UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

HOCHSCHULE HEILBRONN

# Konzeption und Entwicklung einer robotergestützten und ultraschallbasierten Lokalisationskontrolle

## Master-Thesis

zur Erlangung des akademischen Grades
Master of Science (M.Sc.) im Studiengang
Medizinische Informatik

vorgelegt von

Peter Karl Seitz

27. Februar 2018

**Referent:** Prof. Dr. Rolf Bendl

**Korreferent:** Dr. Christoph Maier

**Betreuer:** Prof. Dr. Rolf Bendl

# Zusammenfassung


| | |
|---|---|
| **Titel:** | Konzeption und Entwicklung einer robotergestützten und ultraschallbasierten Lokalisationskontrolle |
| **Autor:** | Peter Karl Seitz |
| **Studium:** | Medizinische Informatik |
| **Abschluss:** | Master of Science (M.Sc.) |
| **Universität:** | Hochschule Heilbronn & Universität Heidelberg |
| **Datum:** | 27.2.2018 |



Mit Hilfe der Image Guided Therapy wird versucht die Bestrahlung von Tumoren mittels Bildgebung zu verbessern und die Nebenwirkungen durch die Bestrahlung für den Patienten zu minimieren. Diese Arbeit verfolgt den Ansatz Ultraschall als echtzeitfähige Bildverarbeitungsmodalität zu nutzen und darüber eine Lokalisationskontrolle von Tumoren während der Bestrahlung zu ermöglichen.
Ziel dieser Arbeit war die Entwicklung und Implementierung eines Gesamtkonzeptes zur ultraschallbasierten und robotergestützten Lokalisationskontrolle. Ergebnis der Arbeit ist eine auf dem medizinischen Bildverarbeitungsprogramm MITK basierte Applikation. Diese erlaubt es ein 2D Ultraschallbild über ein Tracking-System oder den Roboter zu referenzieren und mit registrierten Planungsdaten überlagert darzustellen. Die Darstellung ist dabei in quasi Echtzeit sowohl in 2D als auch in 3D möglich. Zur Registrierung wurde ein optisches Tracking-System verwendet, welches über einen eigens neu entwickelten Filter mit dem Roboter verknüpft werden kann. Weiter lässt sich aus der Applikation heraus der Roboter steuern und es können automatisierte Scanverfahren genutzt werden, um mit Hilfe eines 2D Ultraschallkopfes ein 3D Ultraschallbild zu erstellen. Die Anwendung knüpft an bestehenden Funktionen an und erlaubt es künftigen Nutzern die neu erstellten Komponenten auch getrennt voneinander weiter zu verwenden. Dazu gehören Filter für die Zuordnung der Ultraschallbilder, sowie ein Filter zum Kombinieren von verschiedenen Tracking-Systemen, als auch die Möglichkeit den Roboter zu nutzen.

# Abstract


| | |
|---|---|
| **Title:** | Conception and development of a robot-assisted and ultrasound-based localization control |
| **Author:** | Peter Karl Seitz |
| **Subject:** | Medical Informatics |
| **Degree:** | Master of Science (M.Sc.) |
| **University:** | Heilbronn University & University of Heidelberg |
| **Date:** | Februar 27,2018 |

Image Guided Therapy attempts to improve the radiation treatment of tumors by use of imaging process and tries to minimize the side effects for the patient. This work pursues the approach of using ultrasound as a real-time capable image processing modality and thereby enabling a localization control of tumors during irradiation. The aim of this work was the development and implementation of an overall concept for ultrasound-based and robot-assisted localization control. The result of the work is an application based on the medical image toolkit MITK. It allows a 2D ultrasound image to be referenced via a tracking system or by use of the robotposition and to be superimposed with registered planning data. The presentation in two dimensions or three is approximately possible in real time. For registration, an optical tracking system was used, which can be linked to the robot via a newly developed filter. Furthermore, the robot can be controlled from within the application and automated scanning processes can be used to create a 3D ultrasound image with the aid of a 2D ultrasound head. The application follows existing functions and allows future users to continue using the newly created components separately from each other. These include filters for the assignment of ultrasound images, as well as a filter for combining different tracking systems, as well as the ability to use the robot.

# Danksagung

An dieser Stelle möchte ich mich bei meinem Betreuer Herr Prof. Dr. Bendl für die Ratschläge, Feedback und Motivation bedanken. Dieser Dank gilt genauso Dr Christoph Maier.

Außerdem möchte ich Beatrice Baumann für die aufgewendete Zeit als Probandin und für das Korrekturlesen danken.

Weiter Dank gilt meiner Familie, die mir immer zur Seite stand und mir das Studium ermöglicht hat.

# Inhaltsverzeichnis

# Tabellenverzeichnis



# List of Listings



# Abkürzungsverzeichnis

| | |
|---|---|
| **CIRC** | Kurvenbewegung |
| **CT** | Computer Tomographie |
| **FRE** | Fiducial Registration Error |
| **IGRT** | image-guided Radiation Therapy |
| **ITK** | Insight Toolkit |
| **LIN** | Linearbewegung |
| **MITK** | Medical Imaging Interaction Toolkit |
| **MLC** | Multi Leaf Collimator |
| **MRT** | Magnet Resonanz Tomographie |
| **MTA** | Medizinisch Technische Assistent |
| **NDI** | Northern Digital Inc. |
| **PTP** | Point to Point |
| **ROS** | Robot Operating System |
| **TCP** | Tool Center Point(z.B. Spitze des Ultraschallgerätes) |
| **VTK** | Visualization Tooolkit |

# Abbildungsverzeichnis

# 1 Einleitung

## 1.1 Motivation

Heutige Bestrahlungstechniken erreichen eine hohe Genauigkeit. Durch einen Multi Leaf Collimator (MLC) lässt sich der Bestrahlungsstrahl definieren und somit Tumore exakt umranden. Diese Technik lässt sich z.B. ideal bei Hirntumoren anwenden. Die Zielstruktur kann direkt fixiert werden und die Planungssituation lässt sich genau nachstellen.
Anders sieht es bei der Bestrahlung von Tumoren im Abdomen aus. Atmungsinduzierte Organbewegungen schränken die Wirkungsweise externer Fixierungssysteme deutlich ein. Bei konventionellen Bestrahlungstechniken kann eine ausreichende Dosisapplikation im Tumor oft nur dadurch sicher gestellt werden, dass das Zielvolumen nicht nur den Tumor einschließt, sondern um zusätzliche Margins erweitert wird. Dadurch wird das umgebende gesunde Gewebe stärker belastet und das Risiko für Nebenwirkungen steigt.

In einer Phantom-Studie konnte gezeigt werden, dass Bewegungen die mit Hilfe einer Ultraschallbildgebung erfasst werden, zur Ansteuerung von MLCs verwendet werden können [1]. Die Herausforderung bei der Akquisition von Ultraschallbildern während einer Bestrahlung besteht darin, dass der Ultraschallkopf nicht von einem Untersucher sondern durch spezielle Halterungen positioniert und gehalten werden muss. Rigide Halterungen können die Anforderungen der Ultraschallbildgebung und der Strahlentherapie oft nicht erfüllen. Damit die Ankopplung des Schallkopfes an die Patientenoberfläche und damit die Bildgebung sichergestellt ist, muss die Halterung möglichen Bewegungen des Patienten (Atmung) folgen. Gleichzeitig darf der Schallkopf und die Halterung nicht vom Strahlenfeld erfasst werden. Sie sollten auch die Auswahl möglicher Einstrahlrichtungen nicht oder nur wenig einschränken. In eigenen Vorarbeiten [2, 3] konnte gezeigt werden, dass ein Ultraschallkopf durch einen Roboter geführt werden kann, der die Bewegungen der Patientenoberfläche erfasst und sicherstellt, dass der Schallkopf immer mit konstanter Kraft ankoppelt und dadurch eine kontinuierliche Bildgebung möglich wird. Damit ist eine erste Hürde zur Kontrolle der Patientenlagerung in der Strahlentherapie genommen. Eine weitere wichtige Voraussetzung ist es, Bewegungen, die in den Ultraschallbildern wahrgenommen werden, auf die anatomische Situation bei der Therapieplanung zurückzurechnen. Relativ dazu könnte dann eine Kompensation der Patienten- oder Organbewegung berechnet und zur Steuerung des Linearbeschleunigers bzw. der Patientenliege eingesetzt werden. In der vorliegenden Arbeit wird eine Methodik entwickelt, die es künftig ermöglichen soll, Ultraschallbilder mit den zugrunde liegenden Planungsbildern zu registrieren.

## 1.2 Ziele

Um Bewegungen, die mit Hilfe von Ultraschallbildern detektiert werden, relativ zur Planungssituation ausdrücken zu können, ist die exakte Lage der Ultraschallbilder

im Raum bzw. im Koordinatensystem der Planungsdaten zu bestimmen. Eine Positionsbestimmung des Ultraschallbildes im Raum kann entweder durch die Pose des Roboterarms erfolgen oder durch den Einsatz eines zusätzlichen Tracking-Systems.

Ziel dieser Arbeit ist ein Konzept zu entwickeln, dass diese Zuordnung von Positionsinformation und Ultraschallbild ermöglicht. Da der Roboter mit der Halterung rigide verbunden ist, könnte dieser als Tracking-System verwendet werden. Da es hierfür noch keine Umsetzung gibt, soll eine entsprechende Implementierung und Überprüfung eines solchen Systems erfolgen. Das Führen von Hand des Roboters an Positionen ist generell möglich. Um damit eine punktbasierte Registrierung durchführen zu können aber zu ungenau. Es soll dafür zusätzlich ein optisches Tracking-System verwendet werden. Damit dennoch die Daten des Roboters genutzt werden und keine Sichtverbindung nach der Registrierung notwendig ist, sollen die beiden Systeme kombinierbar sein. Anhand der Tracking-Daten soll dann die Zuordnung der Positionsinformation mit dem Bild erfolgen, dazu gehört eine Kalibrierung von beiden Systemen. Neben dem Konzept soll auf eine spätere praxisnahe Verwendung geachtet werden. So soll die Zuordnung der Positionsinformation in quasi Echtzeit erfolgen und dem Nutzer visuell dargestellt werden. Darüber hinaus soll auch eine überlagerte Darstellung des Ultraschallbildes mit den registrierten Planungsdaten erfolgen. Da die MTAs der Strahlentherapie nicht zwangsweise Erfahrung mit Ultraschall haben und das Platzieren des Roboters schwierig ist, soll es eine Möglichkeit geben ein 3D Volumen, automatisiert mit dem Roboter zu erstellen. Weiter soll mit einem solchen 3D Volumen der Nachweis der richtigen Zuordnung erbracht werden.

## 1.3 Vorgehensweise und Struktur der Arbeit

In Kapitel 2 Grundlagen erfolgt zunächst ein Gesamtüberblick über den Stand der Technik. Anschließend erfolgt eine Beschreibung des möglichen Endsystems anhand eines Anwendungsszenarios 2.1, welche zusätzlich eine Darstellung des späteren Versuchsaufbaus enthält. Es folgt eine Beschreibung der verwendeten Hardware und Softwarekomponenten sowie die notwendigen Grundlagen im Kontext der späteren Anwendung und Programmierung. Da es zu dieser Arbeit einige eigene Vorleistungen gibt, wurde in Kapitel 3 Methodik zunächst eine Evaluation der Ausgangslage (3.1) beschrieben. Diese bildet zusammen mit einer Anforderungsanalyse (3.2) die Basis für die Entwicklung eines Gesamtkonzeptes (3.3). Zur Umsetzung des Gesamtkonzeptes wurden fehlende Komponenten etwickelt. Die Strategie hierbei und das Vorgehen werden anschließend gezeigt. In Kapitel 4 Ergebnisse wird die Umsetzung und das resultierende Gesamtsystem erläutert und dessen Funktionalität anhand von mehreren Tests bewiesen. Kapitel 5 Diskussion bildet den Abschluss der Arbeit und enthält neben einer Zusammenfassung der Arbeit einen Ausblick, wie sich die Ergebnisse der Arbeit zukünftig erweitern lassen.

# 2 Grundlagen

Der Einsatz von Ultraschall zur Lagekontrolle wurde bereits 2002 untersucht [4]. Der Schallkopf wurde mit einer starren Halterung am Patienten positioniert. Die starre Positionierung hat den Nachteil, dass bei Bewegungen des Patienten (Atmung) die Ankopplung des Schallkopfes nicht mehr gewährleistet ist, außerdem schränkt die fixe Schallkopfhalterung die möglichen Einstrahlungsrichtungen erheblich ein, da ein solches System dem Bestrahlungsstrahl nicht ausweichen kann und dieser nicht durch die Halterung strahlen darf. Man benötigt somit ein flexibleres System um die Patientenbewegungen kompensieren zu können. Eigene Vorleistungen konnten zeigen, dass sich hierfür ein Leichtbauroboter nutzen lässt [2]. Die Ergebnisse, der verwendete Roboter und die notwendigen Grundlagen für die in dieser Arbeit hinzukommenden Aspekte befinden sich in Abschnitt 2.2. Dazu zählt insbesondere die Funktionsweise der Atemkompensation 2.2.3. Auch andere Arbeiten untersuchen die Verwendung eines Roboters zur Akquisition von Ultraschallbildern [5, 6, 7, 8, 9].
Neben dem Roboter und dessen Steuerung wird eine Anwendung benötigt, die diese verwaltet und die Verbindung mit anderen Komponenten herstellt. Da die Bestrahlung anhand von CT-Daten erfolgt, sollte eine spätere Anwendung in der Lage sein solche medizinischen Bilddaten von verschiedenen Modalitäten darstellen und verarbeiten zu können. Medical Imaging Interaction Toolkit (MITK) ist ein medizinisches Bildverarbeitungsprogramm, welches diese Funktionen besitzt und sich über Plugins erweitern lässt. Dieses wurde in eigenen Vorarbeiten um eine Einbindung des Roboters erweitert [3]. Die dafür notwendigen Kenntnisse stammen aus der Publikation von Tauscher[10]. Die notwendigen Grundlagen für den Umgang mit MITK und dessen Entwicklung wird in Abschnitt 2.5 erklärt. Das MITK bietet Möglichkeiten sowohl Ultraschallgeräte als auch Tracking-Systeme einzubinden. Das später verwendete Ultraschallsetup wird in Abschnitt 2.3 erklärt. Damit anhand der Ultraschallbilder ein Rückschluss auf Positionen von darin enthaltenen Strukturen gezogen werden kann, benötigt es eine Referenzierung mit einem Referenzsystem. Hierzu lassen sich Tracking-Systeme nutzen. In dieser Arbeit wird später ein optisches Tracking-System verwendet, die Grundlagen hierzu finden sich in Abschnitt 2.4. Ultraschall im Kontext einer Navigation ist bereits im MITK integriert [11]. Eine direkte Verwendung der hier entwickelten Klassen ist aber nicht möglich. Es werden hierbei die segmentierten Objekte in Relation zum Ultraschallbild dargestellt. Die Basis befindet sich dabei an der Spitze des Ultraschallkopfes. Es liefert aber einen guten Vergleich wie ein Ultraschallgerät mit einem Tracking-System im MITK verknüpft werden kann. Lissek [12] liefert bereits einige Vorarbeiten für die Kombination von Ultraschallbildern und optischem Tracking-System im MITK. Weiter enthält diese Arbeit auch Ansätze für das Erstellen von 3D Ultraschallbildern. Außerdem enthält dessen Arbeit eine Kalibrierung der in dieser Arbeit verwendeten Halterung nach Muratore et al. [13].
Der generelle Nachweis ob sich anhand von Ultraschall eine verbesserte Bestrahlung erzielen lässt wurde in mehreren Studien getestet [1, 14, 15, 16, 17, 18, 19]. Ipsen et al. [20] untersucht in einer Phantomstudie in der Strahlentherapie die dosimetrischen Konsequenzen einer Bewegungskompensation mit Hilfe eines MLC auf Basis

von Ultraschallbildern. Teilweise wurden bereits auch schon Versuche mit einem Linac und einem eigens entwickelten Roboter durchgeführt [21]. Salcudean [6] bringt in seinem Fazit den Vermerk, dass eine 3D Ultraschallakquisition über ein solches Robotersystem möglich sein sollte und wünschenswert wäre. Auch für die Registrierung von Ultraschallbildern und Planungsdaten gibt es Ansätze. Ein Beispiel für die intraoperative Registrierung von Ultraschalldaten mit MRT-Bildern liefert die Dissertation von Dekomien [22].

## 2.1 Anwendungsszenario

Ziel der zukünftigen Endanwendung ist eine Positions-und Lagekontrolle in der Strahlentherapie mittels Ultraschall zu realisieren. Ein Szenario könnte daher wie folgt aussehen. Ein Patient mit einem Tumor im Abdomen soll bestrahlt werden. Die Planung hierfür erfolgt auf vorher gewonnenen Computer Tomographie (CT)-Daten und oder Magnet Resonanz Tomographie (MRT)-Daten. Gegebenenfalls sogar mit einem zeitlich aufgelösten MRT. Standardmäßig wird der Patient über äußere Tattoos oder z.B. Masken- oder andere Fixierungssysteme positioniert. In der image-guided Radiation Therapy (IGRT), nach wie vor Ausnahme, weil aufwendig, wird die Position mit Hilfe von 2D Röntgen-Projektionsbildern, mit In-Room-CTs oder mit Cone-Beam-CTs kontrolliert. Dabei werden die Kontrollaufnahmen mit den Planungsdaten registriert, um zu erkennen, ob bzw. wie der Patient verschoben oder neu gelagert werden muss. In ganz wenigen Einrichtungen (weltweit) gibt es seit kurzer Zeit auch MR-Linac Kombinationen, die die Positionskontrolle auf Basis von MRT-Bildern erlauben sollen. Hier Beginnt der Unterschied zu dem neu entwickelten Ansatz. Es erfolgt eine Registrierung ohne Strahlenbelastung über äußere Landmarken. Diese werden jedoch nur für eine grobe Positionierung des Planungsdatensatzes verwendet, da diese Registrierung keine innere Anatomie abbilden muss. Vor der eigentlichen Bestrahlung wird nun ein Ultraschallkopf, gehalten von einem Roboter, an einer passenden Stelle am Patienten positioniert. Passende Stelle bedeutet in diesem Fall, dass auf dem Ultraschallbild die Bewegung des Tumors direkt oder indirekt beobachtet werden kann. Indirekt meint hierbei zum Beispiel das Detektieren von Strukturen neben dem Tumor. Durch Scanfunktionen des Roboters kann ein größerer Bereich geschallt werden und anhand von gewonnenen 3D Ultraschallbildern eine präzisere Registrierung der Planungsdaten erfolgen. Der Roboter kompensiert während der gesamten Zeit fortan die Atembewegungen des Patienten. Im Fehlerfall, z.B. bei Panik des Patienten wird eine Sicherheitsbewegung durchgeführt, die den Roboter vom Patienten weg bewegt und diesen nachgiebig macht. Das bedeutet die Bewegung wird nicht erzwungen, wenn etwas im Weg steht, wird zuerst dem nachgegeben. Über ein Tracking-System oder den Roboter selbst erfolgt die Zuordnung der Position mit dem Ultraschallbild. Dieses lässt sich in das Referenzkoordinatensystem des Planungsdatensatzes bringen.

Über die Auswahl der Schnittebene im CT-Bild oder manuell mittels Joystick lässt sich die Ausrichtung und Position des Ultraschallkopfes verändern. Zur weiteren Lageüberprüfung kann auch eine weitere 3D Ultraschallaufnahme erfolgen. Hierzu kann eine automatische Scanfunktion genutzt werden. Der Roboter fährt dabei eine definierte Bewegung ab, während parallel die Ultraschallbilder aufgenommen werden.
Danach kann die eigentliche Bestrahlung erfolgen. An diesem Punkt könnten verschiedene Ansätze eine genauere Bestrahlung ermöglichen. Die einfachste Variante wäre das Gating. Über das parallel in Echtzeit verarbeitete Ultraschallbild könnte ein Gating erfolgen, dass den Bestrahlungsstrahl startet und stoppt wenn der Tumor an der richtigen Position ist. Komplexere Ansätze wären z.B. ein permanenter Abgleich des Ultraschallbildes mit den Planungsdaten und eine automatische Anpassung der Bestrahlung anhand der Daten des Ultraschallbildes.

### 2.1.1 Versuchsaufbau

In dieser Arbeit kann der im vorigen Abschnitt vorgestellte Ablauf nicht eins zu eins umgesetzt werden. Es wurde daher ein Versuchsaufbau geschaffen, der es ermöglicht wichtige Teile der Arbeit nicht nur theoretisch zu entwickeln, sondern auch praktisch zu testen. Dazu gehört der verwendete Roboter, das Ultraschallgerät, ein optisches Tracking-System zur Registrierung und ein Ultraschallphantom mit dessen CT-Daten. Abbildung 2.1 zeigt das komplette Setup, dass während der Arbeit genutzt wurde.

## 2.2 Grundlagen Roboter

Der Roboter bildet eine Kernkomponente in der späteren Anwendung. Er benötigt hierzu Fähigkeiten um mit dem Patient zu interagieren. Normalerweise dürfen Roboter nur in einem abgesperrten Bereich operieren, die Verwendung eines Roboters mit Kontakt zu einem Menschen stellt somit keine Trivialität dar. Es gibt Richtlinien, an die man sich halten muss, z.B. ISO/TS 15066. Diese beschreibt im wesentlichen welche Voraussetzungen bestehen müssen, damit kein Mensch verletzt wird. Gerade im Umgang mit Patienten, die durchaus im Schnitt älter und gebrechlicher sind, als die jungen verwendeten Probanden, besteht äußerste Vorsicht.

### 2.2.1 Verwendeter Roboter

Ein normaler Industrieroboter erfüllt die Anforderungen für das gesuchte System nicht. Es wurde daher ein Leichtbauroboter mit den Fähigkeiten für die Mensch-Roboter-Interaktion gewählt. Es handelt sich um einen KUKA lbr iiwa 7 800, einem Leichtbauroboter mit 7 Achsen und einer Kraftschätzung. Dieser erlaubt es dadurch auf einwirkende Kräfte zu reagieren und seine Achsen ohne Veränderung des Werkzeugs umzuorientieren. Für die Programmierung des Roboters steht eine eigene Programmierumgebung Sunrise OS, basierend auf Eclipse, bereit. Die Programmiersprache ist

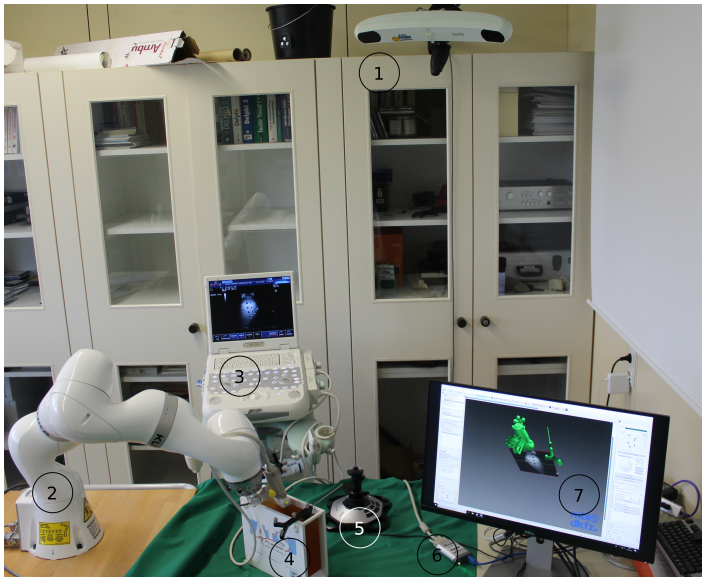



Abbildung 2.1: Versuchsaufbau mit Phantom: (1) optisches Tracking-System NDI Polaris, (2). Roboter Kuka Lbr iiwa 7, (3) Ultraschallgerät Esaote MyLabFive, (4). Ultraschallphantom ATS Mehrzweckphantom Typ 539, (5). Joystick, (6) FrameGrapper, (7). MITK Workbench auf dem Steuerungscomputer.

Java und es gibt eine vorhandene Library um die Funktionen des Roboters nutzen zu können. Alternativ kann auch ein Robot Operating System (ROS) verwendet werden, dieses bietet aber keinen vollständigen Zugriff auf den Roboter. Um möglichst alles nutzen zu können wurde daher mit der Java-API implementiert.

### 2.2.2 Bewegungsarten

Bei der Roboterprogrammierung lassen sich folgende Bewegungsformen nutzen:

- **Point to Point (PTP):** Bei einer PTP Bewegung wird das Anfahren eines Punktes über den Achsraum des Roboters berechnet. Die Strecke die der Roboter zu diesem Punkt wählt kann daher nicht vorhergesagt werden.
- **Linearbewegung (LIN):** Bei einer linearen Bewegung werden zwei Punkte mit einer Geraden verbunden und so angefahren.
- **Kurvenbewegung (CIRC):** Eine Klassische Kreisbewegung.
- **Spline:** Bei einer Spline-Bewegung werden beliebig viele Punkte, die auf einer Ebene liegen nacheinander angefahren.

Neben der Bewegungsform als solche gibt es weitere Konzepte, die diese beeinflussen. So lässt sich z.B. ein Impedanzmodus zur eigentlichen Bewegung hinzufügen. **Impedanz** beschreibt die Nachgiebigkeit des Roboters in einer oder mehreren Achsen. Zum Setzen von zyklischen Positionen z.B. beim Visual-servoing kann die sogenannte **Smart-Servo**-Klasse genutzt werden. Diese erlaubt es dem Roboter während der Bewegung neue Positionen zu übermitteln. Die eigentliche Bewegung wird dabei überschliffen und das nächste Ziel angefahren.
Dies hier ist nur ein kleiner Ausschnitt der Möglichkeiten des Roboters. Eine detaillierte Beschreibung im Kontext der Arbeit kann in meiner Bachelorarbeit [2] nachgelesen werden. Ansonsten steht weiter die Benutzeranleitung des Roboters zur Verfügung [23, 24, 25, 26, 27].

### 2.2.3 Atemkompensationsbewegung mittels Smart-Servo

Wesentliche Vorleistung meiner Bachelorarbeit [2] war das Entwickeln einer Kompensationsbewegung für die Atembewegungen des Patienten. Die Robotersteuerung bietet hierzu den Impedanzmodus an. Die Bewegung setzt sich aus 3 Komponenten zusammen. Die vollständige Nachgiebigkeit in Schallrichtung, dadurch kann der Roboter beliebig in dieser Achse verschoben werden. Eine aufgeschaltene Kraft in Richtung des Patienten, dadurch wird der Roboter nicht nur Nachgiebig in der Schallrichtung sondern versucht konstant eine einstellbare Kraft aufzubauen und diese selbständig konstant zu halten. Letzte Komponente bildet das Verwenden von Smart-Servo, dass es erlaubt während der Atemkompensation den Schallkopf neu zu orientieren oder zu versetzen. Parallel erfolgt eine Kraftüberwachung des gesamten Systems. Wird eine zulässige Maximalkraft überschritten weicht der Roboter nachgiebig von der Patientenoberfläche zurück und hält nachgiebig seine Position. Das heißt er lässt sich von Hand wegdrücken.

### 2.2.4 Werkzeug Kalibrierung

Zur Bestimmung einer Basis, eines am Flange montierten Werkzeugs, bietet der Roboter eine Messmethode an. Die Translation zur neuen Basis kann mittels Tool Center Point(z.B. Spitze des Ultraschallgerätes) (TCP)-Vermessung berechnet werden. Das zu vermessende Werkzeug fährt dabei einen beliebigen Referenzpunkt aus 4 Richtungen an. Eine Ausgabe am Smartpad (Bedienelement des Roboters) zeigt die Translation und den Berechnungsfehler. Sollte dieser zu groß sein wird empfohlen die Messung zu wiederholen.

### 2.2.5 Hintergrundtask

Sunrise OS bietet eine Klasse für zyklische Hintergrundtasks an. Diese lassen sich z.B. für Kontrollaufgaben parallel zur eigentlichen Anwendung ausführen. So kann z.B. die

Position des Roboters an andere Geräte übermittelt werden unabhängig davon welches Programm auf dem Roboter ausgeführt wird. Möchte man einen solchen Hintergrundtask haben, muss die eigene Klasse nur von *RoboticsAPICyclicBackgroundTask* erben. Über das Smartpad kann ein Hintergrundtask dann später an und ausgeschaltet werden.

## 2.3 Ultraschallsetup

Für die Applikation wurde ein Standard 2D Ultraschallkopf verwendet. Alternativ lässt sich für ein 3D Bild ein 3D Ultraschallkopf verwenden. Ein solcher stand jedoch nicht zur Verfügung. Da die generellen Konzepte zur Zuweisung der Bilder mit Positionsinformationen die selben sind, reicht es aus in einem ersten Ansatz nur einen 2D Ultraschallkopf zu verwenden. In zukünftigen Arbeiten muss geklärt werden ob ein 3D Ultraschallkopf überhaupt zwingend notwendig ist. Um dennoch 3D Ultraschallbilder aufnehmen zu können gibt es verschiedene Freihand Scanverfahren, die aus mehreren 2D Ultraschallbildern ein 3D Volumen erzeugen (Abschnitt 2.3.4).

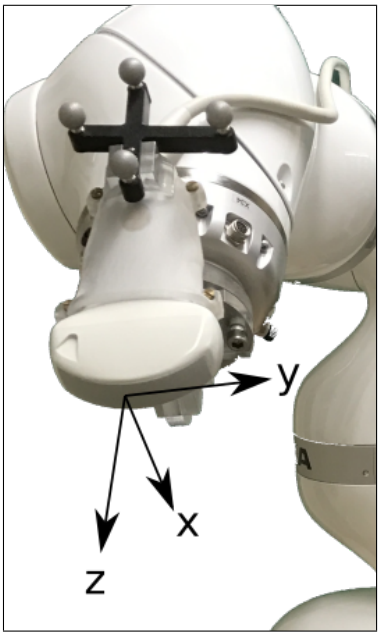
### 2.3.1 Verwendetes Ultraschallgerät

Bei dem verwendeten Ultraschallgerät handelt es sich um das von Esaote produzierte MyLabFive (siehe Abbildung 2.2b), ein mobiles Ultraschallgerät mit Tastatur und Einstellungsknöpfen. Für die Integration in ein medizinisches Bildverarbeitungssystem muss ein FrameGrapper verwendet werden. Daraus ergibt sich, dass in diesem Programm der Bildschirm des Ultraschallgerätees einfach gespiegelt wird. Daher ist es wichtig zu wissen welche Einstellungen am Gerät getätigt wurden und welchen Effekt diese auf das Bild haben. Nachfolgend werden nur die Funktionen erklärt, welche das Bild verändern und eine spätere Maßnahme in der Applikation nach sich ziehen.

- **Tiefe**: Die Tiefe des dargestellten Ultraschallbildes kann von 4 cm bis 30 cm in 1 cm Abständen eingestellt werden. Für jede Tiefe wird aber die gleiche Anzahl an Pixeln für die Darstellung verwendet, dementsprechend muss die Skalierung im Programm später berücksichtigt werden.
- **Winkel**: Beim Curved Array Schallkopf kann der Aufnahmewinkel eingestellt werden. Die Skalierung wird dabei nicht verändert. Es kann aber notwendig sein einen größeren Bildausschnitt zu nutzen oder einen kleineren. Dies spielt später für die Performance eine Rolle.
- **Fokus**: Über die Fokustaste lassen sich bis zu vier Fokuspunkte auswählen. Das Bild bleibt davon unverändert. Wichtig ist jedoch, dass die Bildwiederholrate langsamer wird. Im Kontext mit der Synchronisierung mit dem Tracking-Gerät ist dies somit relevant.

- **Freeze**: Mit der Freezetaste lässt sich das Bild einfrieren. Wird in dieser Zeit der Schallkopf bewegt wird aber weiterhin das gefrorene Bild angezeigt und übermittelt. Erfolgt in der Anwendung eine Positionszuweisung zur gleichen Zeit ist die Bildinformation nicht aktuell und damit falsch.

Für das Ultraschallgerät stehen 3 Ultraschallköpfe zur Verfügung. Ein Curved Array CA431, ein Linear Array LA523 und ein Linear Array PA 230E. Grundsätzlich werden diese für unterschiedliche Einsätze verwendet. Das Curved Array bietet für die ersten Versuche die universellste Möglichkeit. Es wurde daher für diesen eine Halterung entwickelt.




(a) Ultaschallkopfhalterung und Koordinatensystem

(b) Ultrtaschallgerät MyLabFive

Abbildung 2.2: Ultraschallgerät und Halterung des Ultraschallkopfes

### 2.3.2 Halterung

Damit der Roboter und der Ultraschallkopf miteinander verbunden werden können benötigt es eine passende Halterung. Die Verbindung mit dem Roboter ist dabei von zentraler Bedeutung. Je nach Ausrichtung der Halterung werden die Bewegungsmöglichkeiten des Roboters eingeschränkt. In eigenen Vorarbeiten wurde daher eine Halterung entwickelt, welche die Bewegungen des Roboters nach Möglichkeit nicht einschränken [2]. Neben der Position des Schallkopfes ist für die Platzierung des Schallkopfes am Patienten von Hand ein ergonomisches Design notwendig. In Zusammenarbeit mit der Medizinischen Physik Abteilung des DKFZ (Deutsches Krebsforschungszentrum)

wurde hierzu die in Abbildung 2.2a dargestellte Halterung entwickelt. Das dargestellte Koordinatensystem entspricht der Basis für das Tracking-System. Dieses soll später für die Referenzierung der Bilder im Koordinatensystem des Tracking-Systems verwendet werden. Die Halterung wurde hierzu mit einem für das optische Tracking-System notwendigen Rigid-Body versehen.

### 2.3.3 Ultraschallphantom

Zum Testen der späteren Anwendung wurde ein Phantom verwendet. Es handelt sich dabei um ein Mehrzweckphantom Typ 539 der ATS Laboratories. Dieses hat definierte innere Strukturen in Form von mehreren unterschiedlich großen Zylindern, deren Abstände und Größen bekannt sind. Dieses Phantom wurde zur Bestimmung der richtigen Zuordnung verwendet, da dies an realen anatomischen Strukturen ohne Kenntnisse schwer ist. Atembewegungen kann das Phantom nicht simulieren.

### 2.3.4 Verfahren zur 3D Ultraschallakquisition mittels 2D Ultraschallkopf

Für die Akquisition eines 3D Ultraschalls mit einem 2D Ultraschallkopf bieten sich 3 Möglichkeiten. Wie in Abbildung 2.3 dargestellt ist muss der Ultraschallkopf gedreht, geschwenkt oder linear verschoben werden. Damit dabei ein lückenloses Bild entsteht muss die Aufnahmefrequenz, die Anzahl der verwendeten Bilder, der Vorschub oder die Drehgeschwindigkeit entsprechend angepasst werden.

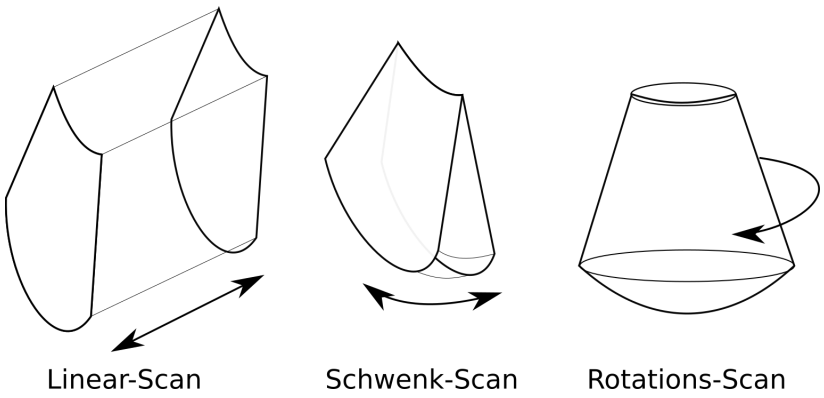



Abbildung 2.3: Scanverfahren zur Erstellung eines 3D Volumen mit einem normalen 2D Ultraschallkopf

## 2.4 Tracking-Setup

Für die Zuordnung der Ultraschallbilder soll ein Tracking-System verwendet werden. Dieses kann später auch der Roboter selbst sein. Zum Vergleich, ob der Roboter sich als Tracking-System nutzen lässt und zur punktbasierten Registrierung wird in der

Arbeit ein optisches Tracking-System verwendet. Nachfolgend wird das verwendete Gerät und ein Programm zum Erstellen von Tracking-Werkzeugen gezeigt.

### 2.4.1 Verwendetes Tracking-System

Bei dem verwendeten System handelt es sich um das optische Tracking-System Polaris von Northern Digital Inc. (NDI). Es nutzt zwei Stereokameras zur Registrierung von optischen Markern auf den Werkzeugen, damit ein Tracking funktioniert ist somit eine ständige Sichtverbindung notwendig. Für die spätere Anwendung existieren zwei Werkzeuge: Ein Pointer für die Registrierung und die in Abschnitt 2.3.2 vorgestellte Halterung für den Ultraschallkopf.

### 2.4.2 Werkzeugkalibrierung mit dem NDI 6D Architekt

Damit das optische Tracking-System anhand von Markern z.B. die Spitze eines Pointers zuordnen kann, benötigt es eine Kalibrierung. Diese beschreibt die Position der Marker im Bezug zu einer gewünschten Basis des Werkzeugs. Die vorhandene Kalibrierung der Ultraschallkopfhalterung war fest im Programmcode hinterlegt und nicht mit der eigentlichen Werkzeug-Datei verknüpft. Das bedeutet es wurde nicht wie in Abbildung 2.2a in Abschnitt 2.3.2 die Spitze des Ultraschallkopfes durch das Tracking-System geliefert, sondern eine andere Basis, die ihren Ursprung in einem der Marker hatte. Damit sich das Tracking-Werkzeug später leichter verwenden lässt, ist es wünschenswert, dass das Tracking-System direkt die Spitze des Werkzeugs liefert und keine weitere Kalibrierungsmatrix für die Transformation zur gewünschten Basis notwendig ist. NDI bietet hierfür den NDI 6D Architekten an. Mit diesem lässt sich das reale Tracking-Werkzeug, genauer dessen Marker, vermessen und eine Basis im Bezug zu diesen erstellen. Über dieses Tool wurde daher die Basis der Halterung wie gewünscht in die Spitze des Ultraschallkopfes gelegt. Ein Screenshot der Anwendung befindet sich in Anhang A.1.

## 2.5 Medical Imaging Interaction Toolkit (MITK)

MITK [28] bietet als open-source basiertes Softwaresystem Funktionen für die Bearbeitung von medizinischen Bildern an. Es baut dabei sowohl auf dem Insight Toolkit (ITK) [29], als auch auf dem Visualization Tooolkit (VTK) [30] auf. Das MITK erlaubt über Plugins neue Funktionen in das Programm zu integrieren. Außerdem können die Plugins von anderen Entwicklern für eigene Zwecke verwendet werden. Um mit der Entwicklung beginnen zu können bieten die Entwickler Tutorials [31] als auch ein detailliertes Entwicklungshandbuch [32] an. Abbildung 2.4 zeigt die Standardansicht ohne zusätzlich geladene Plugins.

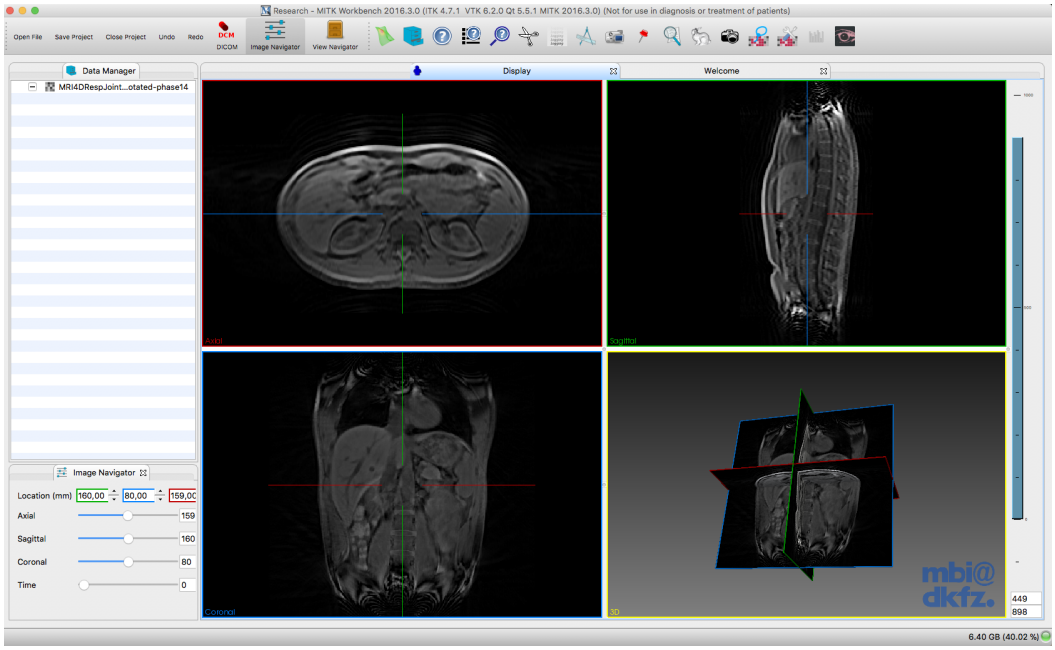



Abbildung 2.4: MITK Workbench. Die Ansicht zeigt einen geladenen MRT-Datensatz. Die Darstellung erfolgt dabei in den 3 orthogonal zueinander liegenden Einzelansichten und einer 3D Ansicht.

### 2.5.1 Grundlagen MITK

Die Gesamtanwendung des MITK setzt sich aus mehreren Plugins und Modulen zusammen. Module beinhalten dabei allgemeinere Funktionen, wie z.B. Filteroperationen, die aus mehreren Plugins genutzt werden. Plugins enthalten oft eine Benutzerschnittstelle, die es dem Benutzer in der MITK-Workbench erlaubt Funktionen des Plugins zu nutzen und darüber z.B. notwendige Filtereinstellungen vorzunehmen. Neben diesem Ansatz bietet MITK viele verschiedene Konzepte und Guidelines für das Programmieren mit der Anwendung und für die eigenen Erweiterungen. So lässt sich für eigene Projekte ein MITK Projekt erstellen. Dieses kann neben dem eigentlichen MITK entwickelt und programmiert werden und enthält die eigenen Module und Plugins. Der folgende Abschnitt bezieht sich im wesentlichen auf das MITK Entwicklerhandbuch [32] und erörtert die Grundlagen im Kontext der Arbeit vereinfacht.

1. **Bildverarbeitung**: Zentrales Konzept, wenn es um Bildverarbeitung im MITK geht, ist das Pipelining. Das bedeutet eine Bildquelle lässt sich mit hintereinandergestellten Filtern bearbeiten und das Ergebnis darstellen.

2. **Filter**: Es gibt verschiedene Arten von Filtern. So lassen sich Bild zu Bild Filter erstellen, als auch Filter die Tracking-Daten manipulieren. Für einen eigenen

Filter muss von dem jeweiligen Basisfilter geerbt werden und die *GenerateData()*-Methode definiert werden, außerdem müssen die Ein- und Ausgänge definiert werden.

3. **Datenquelle**: Um später einen Filter mit etwas zu verbinden braucht es eine Datenquelle z.B. Ultraschallgerät. Eigene Datentypen lassen sich jedoch auch realisieren. Diese müssen dazu von *mitk::BaseDataSource* erben.

4. **MITK Image**: Oberklasse für alle Arten von Bildern. Um auf die Pixel/Voxel des Bildes zugreifen zu können benötigt es einen Accessor (Zugriffsberechtigten). Nachfolgendes Codebeispiel zeigt wie ein Grauwertbild mit 40 x 40 x 40 Pixeln erstellt werden kann. Die letzte Zeile zeigt dann wie ein Accessor für die Pixel zu diesem Bild erstellt wird.

```
1    unsigned int dimensions[] = {  40, 40, 40 };
2    mitk::Image::Pointer image3D = mitk::Image::New();
3    image3D->Initialize(middleImage->GetPixelType(), 3, dimensions);
4    mitk::ImagePixelWriteAccessor<uchar, 3> writeAccess(image3D);
```

Listing 2.1: Codebeispiel zum Erstellen eines Bildes im MITK

5. **Geometrie**: In der Imageklasse wird zusätzlich die Geometrie des Bildes gespeichert. Ohne die Geometrie fehlen wichtige Informationen des Bildes. Sie speichert z.B. folgendes:

   - *Spacing*: Anzahl der mm pro Pixel
   - *Origin*: Den Ursprung des Bildes im Weltkoordinatensystem. Normalerweise liegt der Ursprung in der unteren linken Ecke des Bildes.
   - *Orientation*: Wie das Bild in Bezug zum Weltkoordinatensystem verdreht ist.

   Für die spätere Anwendung war der wichtigste Teil die Zuordnung der Voxel auf Basis der Tracking-Daten. Im folgenden Codebeispiel 2.2 wird daher gezeigt wie sich Indexkoordinaten des Bildes auf Weltkoordinaten umrechnen lassen und andersrum. Zusätzlich wird noch gezeigt wie sich ein Voxel anschließend mit einem Wert versehen lässt.

   Die Geometrie lässt sich als 4x4 Matrix darstellen. Es ist somit möglich homogene Koordinaten zu verwenden. Um auf bestehende Matrixoperationen zurückgreifen zu können muss die 4x4 Matrix in eine *vtk4x4Matrix* konvertiert werden.

6. **Rendering**: Um ein erstelltes Bild anzeigen zu können muss es einfach nur dem *mitk::DataStorage* als *mitk::DataNode* übergeben werden. Ein dort abgelegtes Objekt lässt sich mit Hilfe des *mitk::RenderingManagers* anzeigen.

```
1       mitk::Point3D  worldcoords;
2       itk::Index<3> pixelindex = { { 0, 0, 0 } };
3       itk::Index<3> idx3 = { { 0, 0, 0 } };
4       myimage->GetGeometry()->IndexToWorld(idx3, worldcoords);
5       image3D->GetGeometry()->WorldToIndex(worldcoords, pixelindex);
6       writeAccess.SetPixelByIndexSafe(pixelindex, myimage->
            GetPixelValueByIndex(idx3));
```

Listing 2.2: Beispiel für Pixeloperationen im Bezug auf das Weltkoordinatensystem

7. **Projektstruktur**: Wie bereits erwähnt setzt sich das MITK aus mehreren Modulen und Plugins zusammen. Für die Verwaltung der Plugins und Module wird CMAKE verwendet. Um diese in einem eigenen MITK Projekt nutzen zu können müssen diese in den CMAKE-Dateien entsprechend ausgewählt werden. Abhängigkeiten können in der CMAKE-Datei angegeben werden. Die Reihenfolge wie die Module geladen werden kann dabei eine Rolle spielen. Ein Codebeispiel hierfür ist in Listing 2.3.

```
1 add_subdirectory(glfw-3.2.1)
2 include_directories(glfw-3.2.1/include)
3
4 SET(ADDITIONAL_LIBS ${ADDITIONAL_LIBS} glfw)
5
6 MITK_CREATE_MODULE(JoystickLib
7   INCLUDE_DIRS
8     PUBLIC ${MITK_BINARY_DIR}
9     PRICATE src
10  DEPENDS PUBLIC MitkCore
11  ADDITIONAL_LIBS ${ADDITIONAL_LIBS}
12 )
```

Listing 2.3: Beispiel zum Erstellen eines Moduls mit zusätzlichen Librarys und einer Abhängigkeit zum MITK Core

8. **Micro Services**: Um Funktionalität aus einem Modul oder Plugin in einem anderen Nutzen zu können, wurde im MITK das Konzept der Micro Services implementiert. Möchte man eine Funktion außerhalb des Plugins/Moduls nutzen, so entwickelt man einen Service, der die Funktionalität verwaltet. Dieser lässt sich dann über den Modul-/Plugin-Kontext registrieren oder laden. Ein Beispiel wie so z.B. ein Tracking-Gerät geladen werden kann befindet sich in Listing 2.4.

```
1 void TrackingManager::ConDisconnectOpticalTracking(bool connect) {
2     if (connect) {
3         us::ModuleContext* context = us::GetModuleContext();
4         std::vector<us::ServiceReference<mitk::NavigationDataSource> >
              services = context->GetServiceReferences<mitk::
              NavigationDataSource>();
5
6         for (std::vector<us::ServiceReference<mitk::
              NavigationDataSource> >::iterator it = services.begin();
7             it != services.end(); ++it) {
8             mitk::NavigationDataSource::Pointer currentDevice =
                  dynamic_cast<mitk::NavigationDataSource*>(context->
                  GetService(*it));
9             if (currentDevice.IsNotNull()) {
10                m_OpticalTrackingSource = currentDevice;
11                m_opticalTracking_connected = true;
```

Listing 2.4: Beispiel zum Laden des Trackingdevices in einem eigen Plugin.

### 2.5.2 Verwendete Module

In Abschnitt 2.5.1 wurde erklärt, dass sich verschiedene Module und Plugins nutzen lassen um eine neue Aufgabe zu realisieren. Im Folgenden werden die wichtigsten Module erklärt, die für die Realisierung der Aufgabe direkt oder indirekt verwendet wurden. Diese beinhalten z.B. die benutzten Service Klassen für Tracking-Geräte und Ultraschall.

1. **Ultrasound-Modul**: Das Ultraschall-Modul bietet die wichtigen Klassen wie *mitk::UltrasoundDevice* an. Diese Klasse repräsentiert dabei ein reales Ultraschallgerät und liefert dessen Bilder. Über einen FrameGrapper lassen sich dabei die Bilder des realen Ultraschallgerätes im MITK verwenden. Konfigurieren lässt sich das Ultraschallgerät über das Ultraschall-Plugin.

2. **IGT-Tracking-Modul**: Dieses Modul liefert Klassen zum Verwalten von Tracking-Geräten. Es bietet außerdem Basisklassen für eigene Tracking-Geräte.

### 2.5.3 Verwendete Plugins

Die vorhandenen Plugins bieten Benutzerinteraktionen an um z.B. das Ultraschallgerät und das Tracking-Gerät mit der Anwendung zu verbinden.

1. **Ultrasound-Plugin**: Über das Ultraschall-Plugin[33] lässt sich ein Ultraschallgerät konfigurieren. Vorbedingung ist dabei, dass das Ultraschallgerät (Abschnitt 2.3) über einen FrameGrapper mit dem Computer verbunden ist. Als erstes muss ein neues Gerät erstellt werden. Die Videoquelle muss dabei ausgewählt werden.

Hierfür muss eine Zahl eingegeben werden. Diese ist meistens 1 oder 0 je nachdem ob es z.B. eine verbaute Laptopkamera gibt oder nicht. Unter dem Tab US Image kann das Eingangsbild zugeschnitten werden. Weiter kann die automatische Update-Rate eingegeben werden. Wurde alles richtig eingestellt, wird das Bild dargestellt und die Bildquelle aktualisiert das Bild automatisch. Es kann so in der Workbench als Stream wiedergegeben werden. Das Ultraschallgerät ist außerdem nun als Micro-Service registriert und kann auch von anderen Plugins verwendet werden.

2. **IGT-Tracking-Plugin**: Dieses Plugin vereint mehrere Ansichten und Funktionalitäten. Hauptfenster ist hierbei die *IGTTrackingToolbox*[34]. In ihr lassen sich Tracking-Geräte auswählen. Über einen Menüpunkt lassen sich die zugehörigen Werkzeuge laden oder erstellen. Ist das Tracking gestartet wird die aktuelle Position der Werkzeuge angezeigt. Ist ein Werkzeug außerhalb des Sichtfeldes oder dessen Marker verdeckt wird es rot eingefärbt.
   Unter der Ansicht *MITK-IGT Navigation Tool Manager* [35] können die Werkzeuge auch erstellt und abgespeichert werden. Möchte man ein Bild mit dem Tracking-System registrieren kann die Ansicht *IGT Registration View* [36] verwendet werden.

3. **Measurement-Toolbox-Plugin[37]**:Mit diesem Plugin lassen sich Distanzen in Bildern vermessen. Es wurde verwendet um die Skalierungsfaktor für das Ultraschallbild im MITK zu berechnen.

# 3 Methodik

In diesem Kapitel werden die für das Ergebnis notwendigen Teilschritte beschrieben. Das in 2.1 beschriebene Anwendungsszenario beschreibt welche Funktionen die zu entwickelnde Applikation später haben sollte und wie sich diese in einem Arbeitsablauf widerspiegeln. Da diese Arbeit an eigenen Vorarbeiten [2, 3] anknüpft wurde die Ausgangslage evaluiert (3.1). Der Evaluation folgt die Anforderungsanalyse (3.2). Aus dieser ging hervor, dass für die gesetzten Ziele ein Gesamtkonzept entwickelt werden muss und ein entsprechendes Projekt (Abschnitt 3.4) erstellt werden muss.
Die implementierte Applikation baut wesentlich auf MITK (refsec:grundlagen:MITK) auf. Damit bereits vorhandene Funktionen effizient genutzt werden können wurde sich mit den Grundlagen des MITK auseinander gesetzt und kleinere Test-Programme entwickelt (3.4.3). Für die Zuordnung der Ultraschallbilder im Raum ist ein Tracking-System notwendig. Der Ultraschallkopf ist über eine rigide Halterung mit dem Roboter verbunden. Eine Zuordnung des Ultraschallbildes im Roboterkoordinatensystem müsste somit möglich sein. In Abschnitt 3.5 wurde daher überprüft ob sich der Roboter generell als Tracking-System nutzen lässt. Für die spätere Kontrolle des Roboter-Tracking-Systems wurde ein optisches Tracking-System verwendet. Für dieses gibt es bereits vorhandene Funktionen, die in der eigenen Anwendung verwendet werden sollen. Abschnitt 3.4.3 zeigt wie sich das vorhandene Tracking-System integrieren lässt. Damit ein Tracking-System präzise Daten liefert muss es kalibriert werden. Abschnitt 3.6 beinhaltet sowohl die Kalibrierung des optischen Tracking-Systems (3.6.1) als auch die Kalibrierung des Roboter-Tracking-Systems (3.6.2). Um die beiden Systeme miteinander vergleichen zu können müssen diese in das selbe Koordinatensystem gebracht werden. Der Kalibrierungs-Abschnitt beschreibt daher zusätzlich die Umwandlung von einem Tracking-System in das Andere (3.6.3).
Kernziel der Arbeit war die Verknüpfung von Bildinformation und Tracking-Daten. Hierzu wurde in Abschnitt 3.7.1 ein Filter entwickelt. Zur Kontrolle des entwickelten Filters wurde eine passende Darstellungsform erstellt (Abschnitt3.7.2). Diese zeigt das verarbeitete Bild in der Anwendung an. Ultraschall allein reicht für eine Lagekontrolle nicht aus. Die im Ultraschall erkennbaren Strukturen müssen mit der Planungssituation abgeglichen werden. Die Planung erfolgt in der Regel auf CT-Datensätzen. Eine überlagerte Darstellung von CT-Daten und Ultraschallbildern verbindet die Planungssituation mit der Bestrahlungssituation. In Abschnitt 3.7.4 wurde eine solche überlagerte Darstellung geschaffen. Für eine weitere Plausibilitätskontrolle und als Feature für die Gesamtanwendung sollte aus den verarbeiteten Ultraschallbildern ein 3D Volumen erstellt werden (3.8). Dieses kann später z.B. zur Bestimmung der groben Anatomie oder zum Finden einer idealen Schallebene verwendet werden. Damit das Erstellen eines 3D Ultraschallbildes auch in einem klinischen Setup verwendbar ist, wurde auf eine effiziente Implementierung geachtet und diese überprüft. Während der Bestrahlung hält der Roboter (Abschnitt: 2.2) den Ultraschallkopf. Für die Akquisition eines 3D Ultraschallbildes bietet es sich somit an automatisierte Scanfunktionen zu entwickeln (Abschnitt 3.9). Diese schaffen eine einfachere Handhabung des Systems. Für eine spätere Anwendung könnte auch ein 3D Ultraschallkopf verwendet werden.

Ein solcher stand aber für diese Arbeit nicht zur Verfügung und es ist noch nicht geklärt ob ein solcher für die Lagekontrolle notwendig ist. Weiter sind die Ansätze zur Positionszuweisung der Ultraschallbilder ansonsten identisch. Um dennoch ein Volumen darstellen zu können ist eine entsprechende Funktion essentiell.
Zur Überprüfung der entwickelten Funktionen wurden in Abschnitt 3.10 Tests durchgeführt. Diese beinhalten sowohl Tests mit einem Phantom zur Überprüfung der Funktionalität, als auch Probandentests zum Nachweis der klinischen Tauglichkeit.

## 3.1 Evaluierung der Ausgangslage

Die Arbeit baut auf mehreren Vorleistungen [2, 3] auf. Aus diesem Grund erfolgte eine Evaluation der Ausgangslage. In dieser sollte bestimmt werden welche Funktionen bereits vorhanden sind und welche neu implementiert oder abgeändert werden müssen. Erkenntnisse der Evaluation sind kursiv hervorgehoben und mit einem Pfeil versehen.

### 3.1.1 Architektur des vorhandenen Robotersystems

Die Nutzung eines Roboters als Halter für den Ultraschallkopf ist bereits gelöst [2], sowie die Einbindung in ein medizinisches Bildverarbeitungsprogramm [3]. Hierzu gehören insbesondere Funktionen zur Atemkompensation oder zum Umorientieren des Roboters (2.2.3). Für die definierten Ziele, wie z.B. ein automatisiertes 3D Ultraschall, muss geklärt werden ob die bestehende Architektur und Funktionen ausreichen, ggf. erweitert werden müssen oder ob hier ein neuer Ansatz notwendig ist.
Die rechts schematisch dargestellte Architektur (Abb. 3.1) beschreibt die Ausgangslage der Arbeit. Der Roboter kommuniziert in dieser mit dem MITK über OpenIGTLink. Es wurde eine Kommandohierarchie auf Seiten der Robotersteuerung entwickelt, die es erlaubt implementierte Befehle zu kombinieren. Hierbei nutzen Unter-Kommando-Klassen die eigentlichen Roboterfunktionen und übernehmen die Steuerung. So lässt sich z.B. die Atemkompensation mit einer Umorientierung des Schallkopfes verknüpfen.

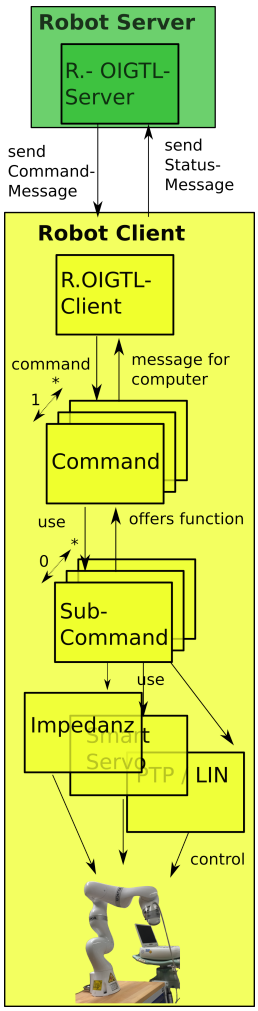



Abbildung 3.1: Architektur der Roboterapplikation

→ *Die vorhandene Architektur auf Seiten der Robotersteuerung kann weiter verwendet werden. Durch die Kombinierbarkeit von Kommandos lassen sich neue Kommandos, wie z.B. Scanfunktionen, realisieren und ohne Aufwand in die bereits verfügbare Roboterapplikation einbauen.*

### 3.1.2 Verbindung mittels OpenIGTLink

Die Verbindung zwischen Roboter und MITK erfolgt über OpenIGTLink. OpenIGTLink ist ein Nachrichtenprotokoll für medizinische Applikationen zum Austausch von Daten. Die Idee dieses Protokoll zu nutzen stammt aus der Publikation von S. Tauscher [10]. In den Vorarbeiten wurden sowohl eigene Nachrichtenklassen, als auch ein eigener Server und Client erstellt, da notwendige Nachrichtenklassen fehlten. Diese Eigenentwicklung entspricht dabei nicht der Grundidee des Protokolls und verhindert eine Verwendung des Roboters aus anderen Systemen heraus. Andersherum können so auch keine Funktionen die von anderen bereitgestellt werden genutzt werden. Verschiedene medizinische Bildverarbeitungssysteme und Geräte unterstützen dieses Protokoll bereits in einer neueren Version. Für das MITK steht so nun ebenfalls eine Einbindung von OpenIGTLink bereit [38]. Die dabei vorhandene neue Version bietet neue Funktionen, welche die eigenen Nachrichtenklassen überflüssig machen. Es wurde daher überlegt die Verbindung mit dem Roboter zu erneuern und dem neuen Protokoll entsprechend anzupassen. Dadurch würde sich der Roboter auch aus anderen Systemen heraus verwenden lassen. Die Einbindung des Roboters wäre somit flexibler und allgemeiner. Die Roboterapplikation ist in Java geschrieben. OpenIGTLink bietet eine Java-Version des Protokolls welche nur eine ältere Version des C++ Version unterstüzt und keine neuen Nachrichtenklassen bietet.
→ *Auf eine Neuimplementierung der Nachrichtenschnittstelle zur besseren Verallgemeinerung kann verzichtet werden. Anderenfalls müsste die komplette Nachrichtenschnittstelle selbst implementiert werden. Die Übertragung zwischen Roboter und MITK kann damit unverändert bleiben und lässt sich ggf. einfacher mit eigenen Nachrichtenklassen erweitern.*

### 3.1.3 Einbindung des Roboters und Ultraschall in MITK

Die Integration des Roboters in das MITK wurde bereits erwähnt. Die Ausgangslage bietet ein MITK-Plugin, mit einer Benutzerschnittstelle zur Steuerung des Roboters. Dieses Plugin beinhaltet zusätzlich Funktionen zur Integration eines Joysticks. Für erste Versuche war diese Implementierung ausreichend. Durch steigende Komplexität der Anwendung ist eine klarere Struktur der gesamten Applikation wünschenswert. Dieses Plugin trennt die Funktionalität jedoch nicht ausreichend und nutzt nicht die Möglichkeiten die MITK bietet.
→*Funktionen des ehemaligen Plugins sollten getrennt werden. Dazu gehört, dass das verwendete Eingabegerät ein eigenes Plugin zur Verwaltung bekommt. Weiter soll es ein Plugin geben um die Robotersteuerung wie bisher nutzen zu können. Zum umorientieren des Ultraschallkopfes soll das Roboter-Plugin dann ein allgemeines Eingabedevice nutzen können.*

Neben der Integration des Roboters gab es ein Plugin mit einem Ansatz für 3D Ultraschall [12]. Damit dieser Ansatz weiter verfolgt werden kann muss eine neuere MITK-Version verwendet werden.
→ *Eine 3D Ultraschallrekonstruktion spielt nicht nur im Kontext der Arbeit eine Rolle. Damit diese Funktion später auch anderen Nutzern zur Verfügung steht soll diese unabhängig von der eigentlichen Applikation sein. Durch den Wechsel der Version und der neuen Struktur der Applikation ist es sinnvoll ein komplett neues MITK basiertes Projekt zu erstellen. Dieses soll klar strukturiert sein und Funktionalitäten trennen. Weiter sollen die Konzepte von MITK genutzt werden. Bisherige Vorleistungen müssen in einem neuen Gesamtkonzept integriert werden.*

### 3.1.4 3D Ultraschall mit dem PLUS-Toolkit

Eine naheliegende Lösung für die Ziele der Arbeit ist die Verwendung des Plus-Toolkits [39]. Dieses bietet Algorithmen zur Kalibrierung von Ultraschallgeräten mit Tracking-Systemen, sowie zur Darstellung eines 2D Ultraschallbildes in 3D, als auch das Berechnen eines 3D Ultraschallbildes. Über das bereits beim Roboter verwendete Nachrichtenprotokoll OpenIGTLink lässt sich dieses Toolkit mit dem MITK verbinden. Da dieses Toolkit wesentliche Funktionen zur Verfügung stellt möchte ich kurz erklären warum dieses dennoch nicht verwendet wurde:

- **Steigende Komplexität**: Durch das verteilte System mit dem Roboter ist ein Debugging der gesamten Applikation schwer. Durch eine zusätzliche Komponente die über eine Nachrichtenschnittstelle kommunizieren muss steigt die Komplexität der Applikation. Dieses birgt Risiken im Fehlermanagement und in der Wartbarkeit der Applikation. Gerade im Umgang mit dem Roboter steigern weitere größere Komponenten die Fehlerwahrscheinlichkeit. Eine eigene Anwendung kann hier klarer kontrolliert werden und besser verwaltet werden.
- **Hoher Initialisierungsaufwand**: Die Verbindung mit dem PLUS-Toolkit bedeutet einen höheren Initialisierungsaufwand. Die Integration des Toolkits bietet einige Fallstricke und ist nicht trivial.
- **Performancevorteile**: Bei einer eigenen Implementierung ist keine zusätzliche Datenkommunikation und deren Verwaltung notwendig. Es könnten somit Performancevorteile bei einer eigenen Entwicklung entstehen. Zusätzlich spielt die Latenz bei einer weiteren Kommunikation eine Rolle.
- **OpenIGTLink**: Keine Mehrfachverwendung von OpenIGTLink, bei dem es nach eigenen Erfahrungen häufiger zu Problemen kommen kann. Zusätzlich sind bereits vorhandene Klassen nicht immer ausreichend dokumentiert.

## 3.2 Anforderungsanalyse

Aus den gegebenen Zielen (Abschnitt 1.2) ergeben sich Anforderungen an die zu implementierende Applikation. Zusätzlich ergeben sich durch die Evaluation der Ausgangslage ( 3.1) Anforderungen an die Applikation. Weiteres Ergebnis der Evaluation ist die Erkenntnis, dass zum Erreichen der Ziele ein MITK basiertes Projekt erstellt werden soll. Dieser Ansatz bietet die beste Möglichkeit bereits vorhandene Funktionen zu nutzen. Die Anforderungen teilen sich daher in zwei Bereiche auf. Der eine Bereich beschreibt die Anforderungen an den Roboter (3.2.1) und der andere die Anforderungen an das neue MITK-Projekt (3.2.2).

### 3.2.1 Anforderungen an den Roboter

Die Anforderungen an den Roboter beschreiben im wesentlichen die Fähigkeit des Roboters im Kontext der Aufgabe verwendet werden zu können. Dazu gehört:

- **Sicherheit:** Der Roboter darf in keinster Form einem Menschen schaden. Dazu gehört eine konforme Programmierung nach ISO/TS 15066. Durch den indirekten Kontakt über die Halterung des Ultraschallgerätes muss eine Kraftüberwachung des Werkzeugs erfolgen.
  - Der Proband/Patient darf nicht eingeklemmt werden. Das bezieht sich vor allem auf Paniksituationen in denen der Proband/Patient versucht aufzustehen.
  - Die Kraft die auf den Proband/Patienten einwirkt muss definierbar sein und beim Überschreiten einer gewählten Grenze muss eine Sicherheitsbewegung durchgeführt werden.
  - Die Sicherheitsbewegung entfernt den Roboter vom Probanden/Patienten und der Roboter gibt dem Probanden /Patienten während dem Ausweichen nach.
  - Es muss mehrere adäquate Möglichkeiten geben in jedem Fall eine solche Sicherheitsbewegung durchführen zu können, z.B. durch entsprechende Tasten.
- **Halterung des Ultraschallkopfes:** Der Roboter soll den Ultraschallkopf während der Bestrahlung halten können.
- **Atemkompensation:** Der Roboter muss während der Kontaktzeit mit dem Probanden/Patienten kontinuierlich dessen Bewegungen kompensieren. Dazu gehört die Einstellbarkeit des Anpressdrucks.

- **Selbständigkeit:** Der Roboter soll unabhängig vom Steuerungscomputer arbeiten können. Das bedeutet, er soll einen Befehl selbständig durchführen und im Falle eines Verbindungsabbruchs die Sicherheitsbewegung durchführen.
- **Positionieren des Ultraschallkopfes:** Der Ultraschallkopf soll sich samt dem Roboter natürlich von Hand an eine gewünschte Stelle platzieren lassen. Anschließend soll dieser sofort mit der Kompensationsbewegung beginnen.

Diese Anforderungen sind nur die wichtigsten Anforderungen an den Roboter. Sie stammen teilweise aus meiner Bachelorarbeit [2, S.15f] und sind im wesentlichen bereits implementiert. Für diese Arbeit ergänzend kommt hinzu:

- **Tracking-Gerät**: Der Roboter soll sich wie ein normales Tracking-Gerät im MITK benutzen lassen. Er soll dabei z.B. die aktuelle Position des Werkzeugs oder des Flanges übermitteln.
- **3D Scanverfahren**: Der Ultraschallkopf lässt sich mit dem Roboter nicht einfach platzieren. Für ultraschallunerfahrene MTAs kann ein Scan der groben Position helfen die richtige Schallebene zu finden. Diese könnte dann auch automatisiert angefahren werden. Der Roboter soll somit einen automatischen 3D Ultraschallscan durchführen können. Wichtig hierbei ist, dass dieser synchron zur tatsächlichen Bildaufnahme im MITK erfolgt. Die Atemkompensationsbewegung soll während der Scanfunktion aktiv sein. Die Bewegung sollte schnell genug sein um eine Aufnahme während des Luftanhaltens eines Probanden zu ermöglichen (10s-20s).

### 3.2.2 Anforderungen an das MITK Projekt

- **Roboter als Tracking-Gerät**: Für die Zuordnung des Ultraschallbildes in 3D soll die Roboterposition verwendet werden können. Dazu soll ein entsprechendes Roboter-Tracking-System entwickelt werden. Über die Benutzerschnittstelle soll dieses angezeigt und verwaltet werden können. Für Testzwecke sollen die Daten des Tracking-Gerätes abgespeichert werden können.
- **Robotersteuerung**: Die Robotersteuerung soll aus einem Plugin mit passender Benutzeroberfläche erfolgen. Dieses soll alle Funktionen enthalten um alle Funktionen des Roboters im Kontext der Arbeit zu verwenden. Dazu gehört das Platzieren von Hand, Ausführen einer Sicherheitsbewegung, Umorientieren des Schallkopfes und Funktionen zum Testen von automatischen Scanfunktionen für das automatisierte akquirieren von 3D Volumen.
- **Verknüpfung Ultraschall mit Tracking-Gerät**: Das Ultraschallbild soll im Raum zugeordnet werden können. Dazu gehört das Verknüpfen des Ultraschallbildes mit einer Positionsinformation. Hierzu soll es möglich sein das Ultraschallbild mit einem beliebigen Tracking-System kombinieren zu können.So soll die

Kombination von Ultraschall mit Roboter- und oder optischem Tracking-Gerät einfach über eine Benutzerschnittstelle realisiert werden.

- **3D Darstellung**: Ein mit Positionsinformation kombiniertes Ultraschallbild soll entsprechend dargestellt werden können. Darunter versteht sich sowohl in einer der 2D Ansichten des MITK als auch in dessen 3D Fenster. Da die Planung der Bestrahlung auf CT-Daten erfolgt soll es weiter möglich sein ein registriertes CT-Bild parallel überlagert darzustellen.
- **3D Ultraschall**: Zum Finden einer Schallebene und als Plausibilitätskontrolle soll es eine Möglichkeit geben um ein 3D Ultraschallbild zu rekonstruieren.
  - *Variabilität 3D Ultraschall*: Es soll möglich sein über die Benutzerschnittstelle die Aufnahmefrequenz und die Anzahl der verwendeten Bilder zu konfigurieren.
  - *Effizienz*: Um nicht nur einen theoretischen Nachweis der Machbarkeit zu haben sollen die Algorithmen zur 3D Rekonstruktion und zur Darstellung in 3D effizient sein. Das bedeutet bei der Darstellung eines Ultraschallbildes in 3D soll die Verknüpfung und Darstellung des Bildes mit der Ortsinformation nicht länger dauern als das nächste Bild geladen wird. Das bedeutet z.B. bei 30 Hz Bildrate darf die Verknüpfung und Darstellung nicht länger als 33 ms dauern. Bei der Akquisition eines 3D Ultraschallbildes darf die Aufnahmezeit nicht länger als die $\frac{Bildanzahl}{Aufnahmefrequenz}$ dauern. Auch das Berechnen des fertigen 3D Bildes darf nicht länger als die dreifache Aufnahmezeit dauern.
- **Einfache Bedienbarkeit**: Durch die Verwendung der vielen einzelnen Komponenten steigt die Komplexität für den Anwender. Die Anwendung sollte daher einfach bedienbar sein um später auch tatsächlich in einem klinischen Setup verwendbar zu sein. Das bedeutet z.B., dass es möglich sein muss Einstellungen vorzudefinieren und z.B. Scanfunktionen automatisiert auf Knopfdruck erfolgen.
- **Unterteilung in Plugins**: Zur besseren Wiederverwendbarkeit sollen Module und Plugins gebildet werden die Funktionalitäten in sich kapseln. Dies soll zudem sicherstellen, dass sich die entwickelten Funktionalitäten unabhängig von einander verwenden lassen. Dadurch sollen diese erweiterbar und veränderbar sein ohne Abhängigkeiten zu erzeugen.
- **Verknüpfung von Plugins**: Funktionen die in einem Plugin/ Modul implementiert sind, die von anderen gebraucht werden, sollen MITK konform über Services verfügbar gemacht werden. Dazu gehört vor allem das Bereitstellen der Funktionalität des Roboters. Für einen automatisierten Scan soll dabei das Plugin für das 3D Ultraschallbild auf einen entsprechenden Service des Roboters zugreifen können.

- **Joystick**: Durch die Aufteilung des ehemaligen Gesamtplugins in dessen Funktionalitäten soll der Joystick ein eigenes Plugin und Modul erhalten. Dieses Modul soll allen anderen Plugins einen Gamecontroller (in diesem Fall den Joystick) zur Verfügung stellen.

## 3.3 Konzept einer Gesamtapplikation

Wie die Evaluation der Ausgangslage (3.1) zeigt, gibt es bereits einige Funktionen die weiter verwendet werden können. Weiter zeigt diese, dass ein neues MITK-Projekt erstellt werden soll um diese Funktionen zu nutzen. Damit ein solches Projekt die gesetzten Ziele erreicht und später weiter entwickelt werden kann, benötigt es ein geeignetes Gesamtkonzept. Da es dieses nicht gibt, musste ein solches entwickelt werden. Das Konzept berücksichtigt dabei die in der Anforderungsanalyse erarbeiteten Punkte (3.2). Das bedeutet, es wurde ein Konzept gesucht, dass neben den eigentlichen Zielen ein Ergebnis erstellt, bei dem einzelne Komponenten ausgetauscht werden können und anderen Nutzern erstellte Funktionen nutzbar gemacht werden. Das Konzept

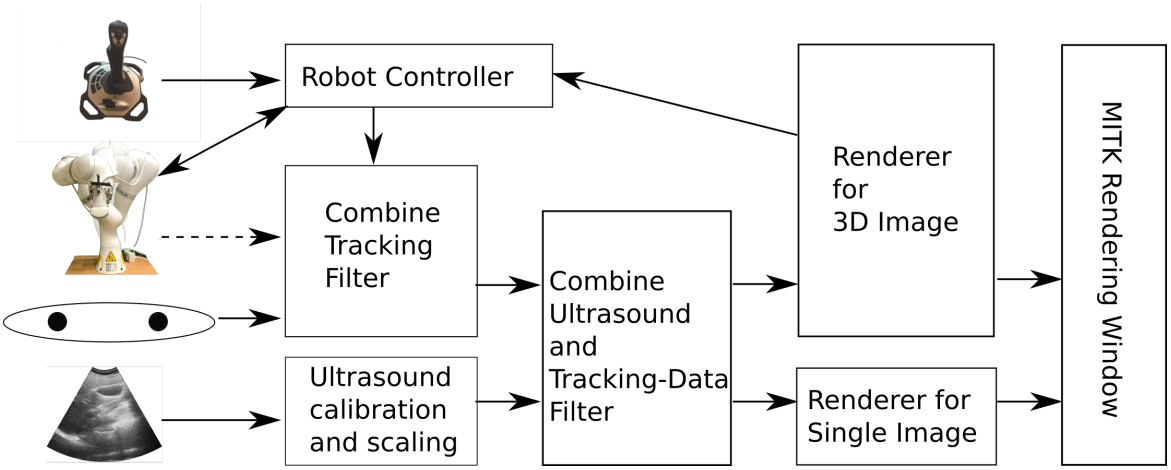



Abbildung 3.2: Grundidee: Es gibt vier Hardwaregeräte die es zu verwalten gilt. Mit verschiedenen Filtern soll aus den Eingängen das 3D Bild errechnet werden. Zur Steuerung des Roboters sind hier der Joystick und der Roboter Controller mit abgebildet. Pfeile deuten die Richtung des Datenflusses an.

der Arbeit ist in Abbildung 3.2 schematisch dargestellt. Auf der Hardwareebene müssen vier Geräte verwaltet werden (Roboter, optisches Tracking-System, Ultraschallgerät und Joystick). Roboter- und optisches Tracking-System sollen dabei in einem ersten Schritt, mit einem Filter, miteinander kombiniert werden. Dies soll z.B. möglich sein um die Vorteile beider Systeme nutzen zu können. Eine Registrierung eines Planungs-CT kann so über das optische Tracking-System erfolgen und die Zuweisung

des Ultraschallbildes über den Roboter. Somit ist keine Sichtverbindung notwendig und kein Werkzeugwechsel am Roboter. Weiter lassen sich die Tracking-Systeme so vergleichen. Parallel zum Tracking erfolgt die Steuerung des Roboters über einen entsprechenden Controller und die Vorverarbeitung des Ultraschallbildes (z.B. Skalierung und Zuschnitt). Anschließend erfolgt die Verknüpfung der Tracking-Daten, aus dem vorher genannten Filter, mit dem vorverarbeiteten Ultraschallbild. Dieses Bild kann abschließend entweder direkt dargestellt werden oder zum Erstellen eines 3D Ultraschallbildes verwendet werden. Die Komponente zum Erstellen des 3D Volumen kann auf die Scanfunktionen des Roboters entsprechend zugreifen. Die Anzeige der Ergebnisbilder erfolgt auf den Standardfenstern des MITK.

## 3.4 Erstellung des MITK-Projekts

Für die Arbeit sollte wie in der Evaluation 3.1 und Anforderungsanalyse 3.2 geschrieben steht eine MITK basierte Applikation erstellt werden. Dieses bildet die Grundlage für die später implementierten Filter und Plugins.
Die MITK Versionsnummer entspricht dabei 2016.11. Ein neueres Release stand nicht zur Verfügung und nach Absprache bot der aktuelle Masterbranch keine großartigen Neuerungen und beinhaltete noch Fehler. Zusätzlich wurden die Plugins und Module für Ultraschall, Tracking, OpenIGTLink, Volumenvisualisierung und Vermessung aktiviert und mitgebaut (siehe auch Grundlagen 2.5.3). Als Entwicklungstool wurde Visual Studio Community 2015, QT 5.6.3 und CMake 3.5.2 verwendet. Eine andere Konfiguration kann zu Fehlern führen! Neben dem so erstellten MITK wurde mit dem Plugingenerator ein neues MITK-Projekt angelegt.

### 3.4.1 Einteilung in Module und Plugins

Aus dem entwickelten Konzept (3.3) leitet sich die Einteilung in Module und Plugins für die Applikation ab. Abbildung 3.3 zeigt eine vereinfachte Darstellung der verwendeten und erstellten Plugins und Module. Die Einteilung der Plugins erfolgt durch die benötigten Benutzerschnittstellen. Diese sind die Steuerung des Roboters, eine Schnittstelle für die neuen Ultraschallfunktionen und am Rande eine Möglichkeit den Joystick zu konfigurieren.

- **Roboter-Plugin:** Dieses Plugin soll notwendige Klassen für die Steuerung und Verwaltung des Roboters enthalten. Außerdem wie bereits erwähnt eine geeignete Benutzerschnittstelle.
- **Ultraschall-Tracking-Plugin:** Dieses Plugin soll die Kernaufgabe der neuen Applikation realisieren. Dazu gehört eine Darstellung des Ultraschallbildes in 3D und Funktionen zum Berechnen eines 3D Ultraschallbildes.

- **Joystick-Plugin:** Dieses Plugin soll später das Handling des Eingabegerätes verbessern.

Aus den notwendigen Plugins leiten sich die Module ab. So lässt sich erkennen, dass der Roboter sowohl aus dem Roboter-Plugin verwendet wird, als auch aus dem Ultraschall-Tracking-Plugin. Das kommt durch die Verwendung des Roboters als Tracking-Gerät und die Möglichkeit einen automatisierten Scan durchführen zu wollen. Es wird somit ein Roboter-Modul benötigt. Dieses muss einen Service besitzen, der den Roboter aus verschiedenen Plugins heraus verwendbar macht. Weiter kann dieses Modul im Rahmen der Arbeit erstellte Filter beinhalten, die unabhängig von den Plugins zukünftig verwendet werden könnten. Das Roboter-Modul soll somit die Basisklassen der zukünftigen Applikation enthalten. Neben dem Roboter-Modul wird auch ein Modul für das Eingabedevice benötigt. Dieses wird sowohl aus dem Roboter-Plugin, als auch aus dem Joystick-Plugin heraus verwendet. Entsprechend des Roboter-Plugins wird auch hier ein passender Service benötigt um die Funktionalität des Eingabegerätes in beiden Plugins zu ermöglichen.
Das Ultraschall-Tracking-Plugin soll zusätzlich auf bereits vorhandene Module wie IGT-Tracking und Ultraschall zugreifen. Durch diese Einteilung ist gewährleistet, dass vorhandene Module genutzt werden und die neu hinzukommenden Funktionen für eine weitere Entwicklung sauber getrennt sind.

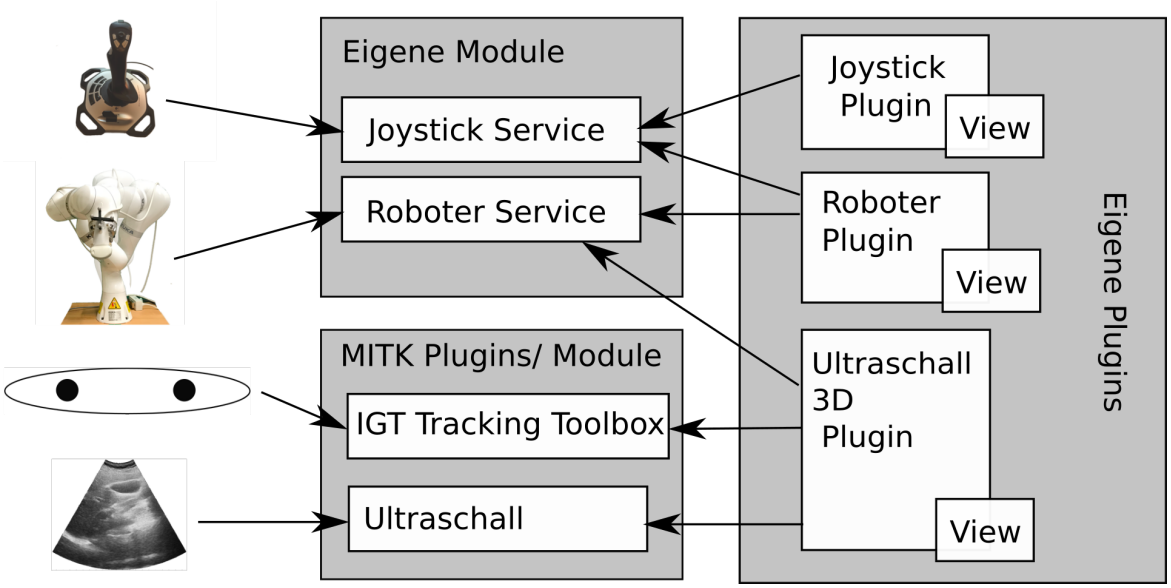



Abbildung 3.3: Einteilung in Module und Plugins. Die Geräte übermitteln dabei ihre Werte entsprechend der Pfeile. Die eigenen Plugins greifen entsprechend der Pfeile auf die Module/Plugins zu.

### 3.4.2 Umstellung der Vorarbeiten auf Micro-Services

Wie im vorhergehenden Abschnitt gezeigt wurden Plugins und Module definiert. Die vorhandene Joystick-Klasse, war ein einfacher Thread, welcher die Daten mittels der GFLW3 Library [40] von dem Joystick auslas. Diese Implementierung erlaubt es aber nicht den Joystick wie gewünscht aus verschiedenen Plugins heraus zu verwenden. Damit die Umorientierung des Ultraschallkopfes, wie vorher, mit dem Joystick funktioniert musste dieser neu implementiert werden.
Um MITK-konform zu arbeiten wurde eine Datenquelle erstellt. Es wurde eine Klasse für die Daten des Joysticks oder eines beliebigen Controllers angelegt, welche von *mitk::BaseData* erbt. Diese Datenklasse kann als Ausgangsvariable der erstellten Datenquelle verwendet werden. Wird die *Update()*-Methode ausgeführt werden die Werte des Joysticks geladen und in den Ausgangsvariablen gespeichert.
Weiter wurde eine Serviceklasse erstellt um die *Joystickdatenquelle* für Plugins nutzbar zu machen. Diese Klasse deklariert dafür das *mitk::ServiceInterface*. Der Joystick soll ohne ein explizites Zutun des Nutzers direkt verwendbar sein. Damit der Service direkt nach Start der Anwendung zur Verfügung steht, wurde ein weiteres Submodul (*JoystickDeviceRegistry*) erstellt, das mittels CMake so konfiguriert ist, dass es sich automatisch mit dem *MITKCore*-Modul lädt. Dadurch steht der Joystick sofort nach dem Start der Anwendung zur Verfügung. Zusätzlich wurde der Service als Singleton implementiert, damit nur eine Instanz des Joysticks existieren kann. Im *Joystick-Plugin* konnte die alte Benutzeroberfläche übernommen werden. Der Zugriff auf die Daten erfolgt nun aber über die Service-Schnittstelle.

### 3.4.3 Zugriff auf bestehende Funktionalitäten des MITK

Die Einteilung in verschiedene Plugins und Module (3.4.1) zeigt, dass neben eigenen Modulen, Funktionen aus bereits bestehenden MITK Modulen und Plugins genutzt werden sollen. Damit eine Kombination von Ultraschall und Tracking-Information möglich ist, soll das Ultraschall-Tracking-Plugin einen Zugriff auf das Ultraschallgerät, das optische Tracking-System und später den Roboter erhalten. Die in den Grundlagen erklärten Plugins (2.5.3) verwalten z.B. das Ultraschallgerät und das Tracking-System. Über Micro-Services lassen sich deren Funktionen aus einem eigenen Plugin heraus nutzen.
Das optische Tracking-System kann im IGT-Plugin über den *mitk::Tracking-Device-Manager* konfiguriert werden. Der Zugriff aus dem eigenen Plugin erfolgt über die *mitk::Tracking-Device-Type-Collection*. Diese wird über den Plugin-Kontext geladen. Auf der Benutzeroberfläche des Ultraschall-Tracking-Plugins wurden Buttons definiert um ein verbundenes Tracking-System im eigenen Plugin zu laden oder zu entfernen. Zusätzlich wurde eine Anzeige erstellt, welche die Werte des Tracking-Systems anzeigt. Intern wurde hierzu eine Manager-Klasse *(TrackingManager)* erstellt. Diese wurde als Thread implementiert um zyklisch die Tracking-Daten des Tracking-Gerätes abrufen

und die Position auf der Benutzerschnittstelle darstellen zu können. Weiter wurde eine Aufnahmefunktion geschaffen um die Daten des Tracking-Systems abspeichern zu können. Dies ist notwendig um spätere Tests durchführen zu können. Die Aufnahmefunktion kann dabei über die Benutzerschnittstelle an und ausgeschaltet werden. Die Daten werden dann in einem Logfile gespeichert.
Das Ultraschallbild wurde ähnlich wie das optische Tracking-System über eine Servicestelle geladen. Die erstellte Manager-Klasse (*TrackingManager*) wurde um eine lade und entferne Ultraschallgerät-Methode erweitert. Somit verwaltet diese Klasse Ultraschallbild und Tracking-System.

## 3.5 Entwicklung eines Roboter-Tracking-Systems

Durch die Verwendung des Roboters zur Atemkompensation stellt sich die Frage ob dieser nicht auch als Tracking-System für das Ultraschallbild verwendet werden könnte. Ein solches Roboter-Tracking-Systems ist im MITK noch nicht vorhanden. Zur Überprüfung der Möglichkeiten eines roboterbasierten Tracking-System musste ein solches erstmal entwickelt werden.

### 3.5.1 Variante 1: Open IGTLink-Tracking-Gerät

Der hier gezeigte Ansatz zeigt wie ähnliche Fragestellungen in der Literatur gelöst und angegangen werden. Diese nutzen für ein beliebiges Tracking-System die bereits erwähnte Nachrichtenschnittstelle OpenIGTLink. Dessen Tracking-Funktionen sind im MITK bereits hinterlegt. Ich erkläre hier, warum dieser Ansatz nicht funktioniert hat und warum man ihn dennoch nicht außer Acht lassen darf. Der später verwendete alternative Ansatz wird im nächsten Abschnitt erklärt.
Über in OpenIGTLink verfügbare Positionsnachrichten können Positionsinformationen geschickt werden. Der Client kann sich mit dem Server verbinden und von diesem die Tracking-Daten erhalten. Über das IGT-Tracking-Plugin kann die Serveradresse und der Port eingestellt werden. Wichtig bei dieser Umsetzung ist, dass das MITK als Client fungiert, der von einem Server die Trackingposition empfängt. Bis jetzt war der Roboter nur als OpenIGTClient implementiert. Um nichts an der Kommunikation zwischen Roboter und MITK zu verändern wurde sich dafür entschieden einen Hintergrundtask in der Robotersteuerung zu implementieren. Gedacht war es, den Roboter als permanenten Server einzurichten, der seine Position an jeden Client schickt, der danach fragt. Das hätte zusätzlich den Vorteil, dass der Roboter unabhängig von der laufenden Applikation seine Position zur Verfügung stellt. Außerdem lassen sich Hintergrundtasks einfach über das Smartpad aktivieren und deaktivieren.
Es wurde daher ein Hintergrundtask erstellt, der als OpenIGTLink Server fungiert und die Werte des Roboters entsprechend schickt. Problem dabei ist, dass die Java Version von OpenIGTLink noch nicht die entsprechenden Klassen für die Schnittstelle im MITK besitzt. Während den Tests dieses Tracking-Systems riss die Verbindung

mit dem Roboter permanent ab. Es konnte nicht herausgefunden werden woran das liegt. Im Verdacht steht die ältere OpenIGTLink-Version des Roboters. Die Unklarheit ob die OpenIGTLink-Java-Version demnächst erneuert werden würde und ob eine Änderung des Systems eine Lösung dieses Problems bedeutet, führten zum zweiten Ansatz.

### 3.5.2 Variante 2: Observer auf dem Roboter

Es besteht bereits eine Verbindung zwischen MITK und dem Roboter. Dieser schickt seine Achsposition zur Darstellung im MITK. Für die alternative Variante wurde die bereits bestehende Verbindung mit dem Roboter genutzt. OpenIGTLink wurde hierzu behelfsmäßig um eine eigene Klasse erweitert, welche die Positionsinformation als Quaternion und Translationsdarstellung abspeichert. Der Observer auf Roboterseite schickt nun sowohl die Achsstellung als auch die Position hintereinander. Auf Seiten des Roboters wurde der bestehende Observer erweitert. Dieser bekommt die aktuelle Position des Werkzeugs und wandelt diese in eine Quaternionendarstellung um. Diese wird benötigt, da im MITK die Basisklassen für Tracking-Systeme Quaternionen zur Repräsentation intern verwenden. Durch die Verwendung der bestehenden Verbindung fehlt ein entsprechendes Tracking-Gerät im MITK. Es wurde daher ein eigenes Tracking-Gerät erstellt. Hierzu wurde entsprechend der IGT-Tracking-Anleitung[1] vorgegangen. Die folgenden erstellten Klassen wurden im *Roboter-Modul* erzeugt. Schematisch und vereinfacht ist das Ganze in Abbildung 3.4 dargestellt. Sie zeigt zusätzlich den Unterschied zwischen der Einbringung des optischen Tracking-Systems und des neu erstellten Roboter-Tracking-Systems.

- *mitkRobotTrackerTypeInformation*: Klasse für die Definition des neuen Tracking-Gerätes und zum Erzeugen eines Tracking-Gerätes.
- *mitkRobotTrackingDevice*: Eigentliche Repräsentation eines realen Tracking-Gerätes. Über die Methode *TrackTools()* erfolgt hierbei das eigentliche Tracking. Diese nutzt die interne Klasse *RobotTrackingDataSource*.
- *RobotTrackingDataSource*: Diese Klasse liefert die Werte vom OpenIGTLink Server an das *mitkRobotTrackingDevice*.
- *mitkRobotTrackingTool*: Da ein Tracking-Gerät meistens mehrere Tools haben kann hier ein Stellvertreter. Real kann der Roboter aber nur seine Position liefern.
- *RobotPoseMessage*: Nachrichtenklasse zur Positionsübermittlung.

Aus dieser Klassenkonstellation ergibt sich folgender umgesetzter Ablauf:

1. Der Roboter schickt die Position als *RobotPoseMessage*.

[1] *http://docs.mitk.org/nightly/IGTHowToImplementATrackingDevice.html*

2. Der *OpenIGTLinkServer* empfängt die Nachricht und leitet sie an den *RobotManager* weiter. Die textitRobotManager-Klasse ist die Basisklasse zur Verwaltung des Roboters. Diese erbt von *RobotTrackingDataSource* und dient so neben der Verwaltung des Roboters nun auch als Quelle für Tracking-Daten.
3. Die *RobotTrackingDevice*-Klasse holt sich die aktuelle Position von der *RobotTrackingDataSource* und stellt die Werte als normales Tracking-Gerät zur Verfügung. Gleichzeitig verwaltet und registriert er das so erstellte Tracking-Gerät.
4. Ein anderes Plugin kann sich das registrierte *RobotTrackingDevice* laden und fortan die Werte davon erhalten. Dies übernimmt später die *TrackingManager*-Klasse aus dem Ultraschall-Tracking-Plugin (siehe auch 3.4.3).

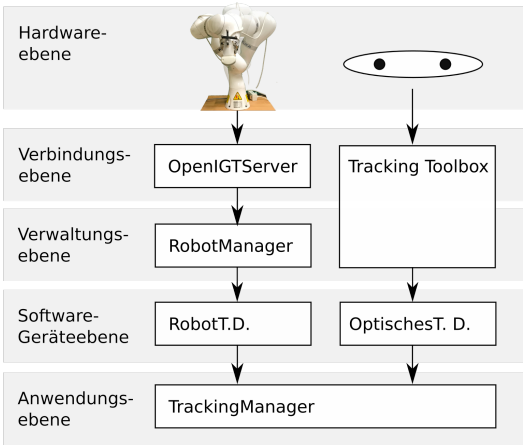



Abbildung 3.4: Die Grafik zeigt, wie der Roboter im Vergleich zum optischen Tracking-System eingebunden wurde.

Damit durch den parallelen Zugriff auf die Tracking-Daten keine Fehler entstehen wurden Mutex verwendet. Beim Schreiben der Werte werden die Ausgangsvariablen entsprechend gesperrt und anschließend wieder freigegeben. Es wurde hierfür die MITK eigene Mutex-Klasse verwendet.

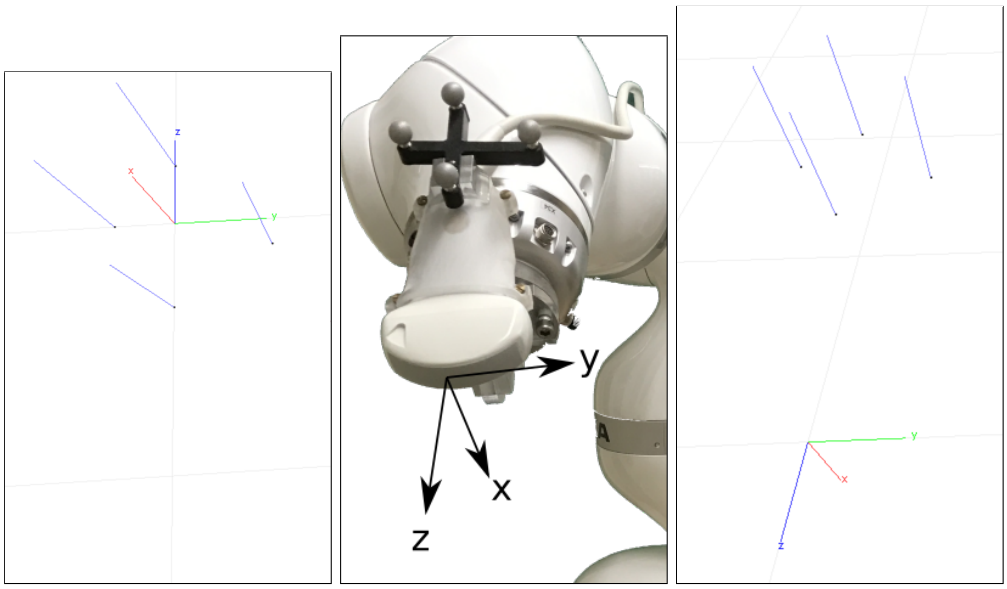
## 3.6 Kalibrierung und Kombination der verwendeten Tracking-Systeme

Die Einbindung des Roboters als Tracking-System alleine reicht nicht aus. Damit ein Tracking-System präzise Daten liefert muss dieses kalibriert sein. Für die spätere Applikation müssen zwei Tracking-Systeme kalibriert werden. Für das optische Tracking-System steht bereits eine Kalibrierung zur Verfügung. Bei dieser musste aber die Art und Weise wie diese Kalibrierung verwendet wird angepasst werden (3.6.1). Auch für den Roboter stand eine erste Kalibrierung zur Verfügung. Diese reichte aus um den

Ultraschallkopf neu zu orientieren oder zu versetzen. Damit beide Tracking-Systeme sich vergleichen und kombinieren lassen müssen diese die selbe Basis verwenden. In Abbildung 3.5b ist zu erkennen wie diese Basis aussehen soll. Sie wurde so gewählt das es einfach ist den Roboter über diese Basis zu bewegen. Zusätzlich liegt das Ultraschallbild später auf y-z-Ebene. Das bedeutet, dass Bild muss nur orthogonal gedreht und versetzt werden.

### 3.6.1 Zuweisung Ultraschallbasis optisches Tracking-System

Wie in Abbildung 3.5b dargestellt liegt die gewünschte Basis für das optische Tracking-System in der Ultraschallkopfspitze. Die Kalibrierung aus den Vorarbeiten[12] entspricht diesem Punkt, jedoch stimmt die Orientierung der Achsen nicht. Weiter wurde diese Kalibrierung fest in dem Programmcode hinterlegt. Abbildung 3.5a zeigt wie die Basis des optischen Systems definiert war. Das bedeutet das Tracking-System liefert die Position in der Mitte der Marker. Wünschenswert wäre es aber wenn das System die Daten der definierten Basis direkt liefert und hierfür später keine weitere manuelle Transformation notwendig ist. Das hat auch den Vorteil, dass eine Repräsentation des Werkzeugs an diese Basis leichter angepasst werden kann.




(a) NDI 6D Architekt Darstellung der alten Basis

(b) Halterung des Roboters mit der gewünschten Basis

(c) NDI 6D Architekt Darstellung der neuen Basis

Abbildung 3.5: Rigid Body der Ultraschallkopfhalterung und die dazugehörige Basis

Die Daten des Tracking-Werkzeugs werden in einer .rom-Datei abgespeichert. Über den NDI 6D Architekten kann das Werkzeug kalibriert werden. Die gegebene Kalibrierung wurde mit der in der .rom-Datei erstellten Basis verknüpft. Die sich daraus

ergebende Transformation wurde über einen Basiswechsel in die neue Basis überführt. Im NDI 6D Architekten wurde ein neues Werkzeug angelegt und die Position der Marker entsprechend der berechneten Transformation neu gesetzt. Abbildung 3.5c zeigt die neue .rom-Datei. Es ist zu erkennen, dass die Basis wie gewünscht definiert ist.
Um eine weitere Kontrolle dieser Transformation zu bekommen sollte ein Tracking-Werkzeug im MITK angelegt werden. Dazu gehört die bereits gewonnene .rom-Datei als auch die grafische Repräsentation des Werkzeugs. Diese fehlte noch. Da die Halterung aus dem 3D Drucker stammt, konnte die Grafikdatei der Halterung als .stl-Datei übernommen werden. Jedoch stimmten die Koordinaten der Grafikdatei nicht mit den gewünschten überein. Durch die neu definierte Basis ist eindeutig beschrieben wie die Halterung im Bezug zur Basis aussieht. Die .stl-Datei wurde hierzu in einem 3D Bearbeitungsprogramm (Blender[2]) entsprechend der Basis neu orientiert und skaliert.
Über den ToolManager des IGT-Plugins wurde dieses neue Werkzeug in das MITK eingefügt. Anhand des vorhandenen Pointers, konnte überprüft werden ob die Orientierung und die Basis des Werkzeugs visuell stimmt. Auf eine Überprüfung der Kalibrierung konnte verzichtet werden, da der Fehler dieser Kalibrierung bekannt war. Der Fiducial Registration Error (FRE) entspricht hierbei 1,275 mm. Da eine genauere Kalibrierung zu zeitaufwendig war wurde auf diese verzichtet.

### 3.6.2 Zuweisung Ultraschallbasis Roboterflange

Wie im vorhergehenden Abschnitt beschrieben ist es wichtig, dass beide Tracking-Systeme die selbe Basis verwenden. Dieses ermöglicht es später mathematisch gesehen die Basen einfach zu wechseln. Für die Kalibrierung des Roboterwerkzeugs stehen mehrere Methoden zur Verfügung. Damit die beiden Basen möglichst identisch sind, wurde sich dafür entschieden die Basis des Roboters mit Hilfe des optischen Tracking-Systems zu vermessen. Abbildung 3.6 zeigt hierzu den Versuchsaufbau. Ein optischer Tracking-Pointer wurde fixiert. Dadurch besitzt dessen Spitze über die gesamte Zeit die gleiche Position. Weiter ist die Basis des optischen Ultraschall-Werkzeugs bekannt. Der eigentliche Ultraschallkopf wurde hierzu aus der Halterung entfernt. Es ist dadurch möglich die Halterung und den Tracking-Pointer im optischen System auf die gleiche Position zeigen zu lassen.
Die verfügbare Vermessungsfunktion des Roboters benötigt einen fixen Punkt. Dieser bildet der Tracking Pointer. Dieser Punkt muss dann aus 4 verschiedenen Richtungen angefahren werden. Zum Abgleich ob sich die Halterung an der richtigen Position befindet wurde geschaut, ob die Position der beiden optischen Werkzeuge identisch war. Waren diese um 0,5 mm genau wurde die Position im Roboter abgespeichert und der Punkt aus einer neuen Richtung angefahren. Mit Hilfe der so bestimmten Positionen berechnet der Roboter die Transformation vom Roboter-Flange selbständig. Der Berechnungsfehler betrug 0,99 mm. Da das Werkzeug rechtwinklig auf dem Roboterflange

[2]https://www.blender.org

montiert ist, musste die Orientierung im Bezug zum Roboterflange nicht vermessen werden. Sie entspricht der gleichen Orientierung wie der Roboterflange selber.


Abbildung 3.6: Versuchsaufbau Kalibrierung Roboter

### 3.6.3 Verknüpfung optisches Tracking-System und Roboter-Tracking-System

Damit eine Umwandlung vom einen Tracking-System in das andere erfolgen kann benötigt es eine Basis-Transformation. Es wurde hier entsprechend des entwickelten Konzepts in Abschnitt 3.3 ein Filter entwickelt, der die beiden Tracking-Systeme kombiniert. Hierzu wurde ein Navigationsdaten-zu-Navigationsdaten-Filter erstellt. Am Eingang des Filters liegt sowohl das optische Tracking-System als auch der Roboter an. Am Ausgang kommt dann der Datenstrom des ersten Eingangs heraus. Intern wird jedoch geprüft ob dessen Daten valide sind, sollten sie es nicht sein, werden die Koordinaten des zweiten Eingangs verwendet und vorher in die andere Basis überführt.
Für die Basistransformation benötigt es eine vorher bestimmte Basistransformationsmatrix. Hierfür wurden homogene Koordinaten verwendet. Da die Basen praktisch identisch sind (Genauigkeit der Kalibrierung vorausgesetzt) kann die Transformation der Basis entsprechend der Abbildung 3.7 einmalig anhand der Position des Werkzeugs erzeugt werden durch:
Die Transformation zwischen Roboterfuß und optischen Tracking-Systemkoordinaten ist gegeben durch $T_{optischNachRobot} = T_{Robot} * T^{-1}_{optisch}$. Daraus ergibt sich, möchte man die Koordinaten des Roboters anhand der optischen Daten wissen, folgende Transformation:
$\rightarrow T_{Robot'} = T_{optischNachRobot} * T_{optisch}$. Andersherum lautet die Transformation:

$\rightarrow T_{optisch'} = T^{-1}_{optischNachRobot} * T_{Robot}$
Grundsätzlich wäre das Arbeiten mit den Roboterkoordinaten sinnvoller. Dieser benötigt keine Sichtverbindung und liefert die Position im Submillimeterbereich genau. Da jedoch später bei der Registrierung mit einem CT-Bild das zugehörige Plugin mit den optischen Daten arbeitet werden hier vorerst die Daten des optischen Systems genutzt und nur ggf. die Daten des Roboters.

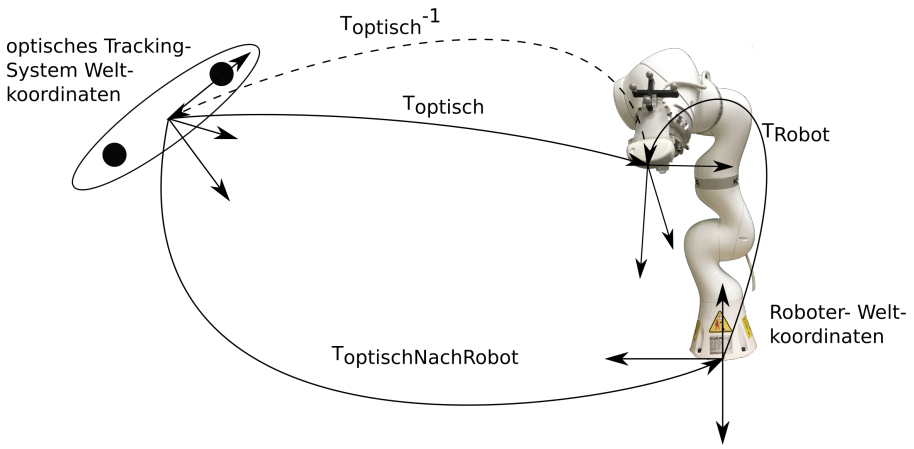



Abbildung 3.7: Tracking-Systeme und ihre Basen. Die Pfeile deuten die Transformation in das entsprechend andere System an.

### 3.6.4 Test der Umwandlung der Tracking-Daten

Die Mathematik allein reicht nicht aus um zu zeigen, dass sich die beiden Systeme kombinieren lassen. Durch die Kalibrierung der einzelnen Systeme, Latenzen und numerischen Probleme bei der Matrizenberechnung muss überprüft werden ob die Transformation auch in der Praxis funktioniert und welche Fehler diese hat. Es wurde eine Messung durchgeführt in der sowohl die Position und die Orientierung des Ultraschallkopfes verändert wurde. Dies erfolgte über die Robotersteuerung mittels Joystick. Die Bewegung erfolgte mit der Maximalgeschwindigkeit des Roboters bei der Atemkompensation. Dadurch kommen ggf. auch Latenzen eher zur Wirkung als bei einer langsamen Bewegung. Der berechnete Fehler ist somit der maximale Fehler. Zum Berechnen des Fehlers wurde der im vorhergehenden Abschnitt erstellte Filter verwendet. Dieser wandelt die Daten des Roboters in Daten des optischen Tracking-Systems um. Anschließend werden beide Daten in einer Datei abgespeichert. Es wurden so 500 Werte in 100 ms Abständen aufgenommen. Es wurde jeweils die Differenz gebildet und der euklidische Abstand berechnet. Das Ergebnis befindet sich in Abschnitt 4.1.1.

### 3.6.5 Ultraschall

Unabhängig von dem Tracking-System muss auch das Ultraschallbild kalibriert werden. Wie in Abschnitt 2.3 gezeigt wurde kann das Ultraschallbild verändert werden. Die Anzahl der Pixel bleibt jeweils aber gleich. Aus Sicht des MITK gibt es daher für das Ultraschallbild nur eine Größe zu kalibrieren. Das sogenannte Spacing. Es sagt aus wie viele mm einem Pixel/Voxel entsprechen. Das Spacing muss verändert werden, wenn z.B. die Tiefenauflösung des Ultraschalls verändert wird. In der Benutzeroberfläche wurde daher eine Schaltfläche erstellt, in der das Spacing direkt eingetragen oder eine vorher abgespeicherte Konfiguration ausgewählt werden kann. Das Spacing lässt sich mit Hilfe des Ultraschallgerätes bestimmen. Mit Hilfe des Vermessungstools kann auf Seiten des Ultraschallgerätes eine Strecke vermessen werden. Durch die Übertragung des FrameGrappers ist die Strecke im MITK ersichtlich. In diesem kann nun die Strecke vermessen werden. Aus dem Quotient der beiden Strecken lässt sich der momentane Skalierungsfaktor ablesen. Die Standardeinstellung des Ultraschallgerätes wurde hierfür auch als Standard in dem Plugin gesetzt. Für andere Einstellungen am Ultraschallgerät lässt sich dieser Wert aber über die Benutzerschnittstelle neu setzen. Die Basis des Ultraschallbildes liegt in der linken unteren Ecke des Bildes. Das bedeutet es wird eine weitere Transformation benötigt um das Ultraschallbild an dem Tracking-Werkzeug entsprechend zu positionieren. Das Bild muss dabei als erstes um die Hälfte in x-Richtung verschoben werden. Dadurch liegt es mittig an der Ultraschallkopfspitze. Anschließend muss es noch in die y-z-Ebene verdreht werden. Diese Transformation wird in einer Matrix gespeichert und lässt sich für die nächsten Abschnitte verwenden (3.7). Diese Kalibrierungsmatrix ist notwendig, da sich die Größe des Ultraschallbildes am Ultraschallgerät ändern lässt. Eine Zuweisung der Tracking-Werkzeuge direkt auf den Ursprung des Bildes (anstelle der Spitze des Ultraschallkopfes) wäre nicht für variable Ultraschallbilder möglich. Hier müsste für jede Tiefeneinstellung ein neues Werkzeug definiert werden. Die Transformationsmatrix wird im Ergebnis gezeigt 4.1.1.

## 3.7 Darstellung des Ultraschallbildes in 3D

Hauptaufgabe der Arbeit ist die Verknüpfung von Tracking-Information mit dem Ultraschallbild. Neben der Verknüpfung der Daten ist auch eine entsprechende Darstellung notwendig. Diese bietet dem Anwender ein visuelles Feedback. Speziell in der 3D Ansicht des MITK werden die Tracking-Werkzeuge bereits angezeigt. Wenn das Ultraschallbild in dieser Darstellung zusätzlich angezeigt wird, lässt sich visuell erkennen ob der Versuchsaufbau und damit das Ultraschallbild rein visuell richtig in der MITK-Workbench dargestellt wird.

### 3.7.1 Ultraschall 3D Filter

Zum Verknüpfen der Daten wurde ein Filter entwickelt. Ideal wäre ein Filter mit zwei Eingängen, einer für Tracking-Daten und einer für das Ultraschallbild. Ausgang wäre dann das mit den Tracking-Daten verknüpfte Bild. MITK stellt aber nur einen Bild zu Bild, Tracking-Daten zu Tracking-Daten- Filter als Basisklasse bereit.
Ähnliche Probleme werden im MITK in anderen Plugins mit dem Tracking-Daten zu Tracking-Daten- Filter gelöst. Es wurde sich daher für den gleichen Ansatz entschieden. Zusätzlich kann der so erstellte Filter mit dem Tracking-Kombinationsfilter gemäß dem Konzept (3.3) verknüpft werden. Das bedeutet der neue Filter erbt von *mitkNavigationDataToNavigationDataFilter*. Somit lässt sich ein Tracking-Gerät direkt mit diesem Filter verknüpfen. Das Ultraschallgerät wurde als zusätzliche Variable eingebunden. Bei diesem Ansatz wird das Bild passiv verändert.
Der Filter lädt dabei zuerst die Daten des Tracking-Gerätes. Die Daten des Tracking-Gerätes werden mit der in Abschnitt 3.6.5 erstellten Kalibrierungsmatrix multipliziert. Anschließend wird die Transformation der Basisgeometrie der *mitk::Image*-Klasse übergeben. Zuletzt wird noch das Spacing entsprechend gesetzt.
Über die Benutzerschnittstelle lässt sich auswählen welches Tracking-Gerät mit dem Filter verbunden werden soll.

### 3.7.2 Renderer für das Ultraschallbild in 3D

Durch den in Abschnitt 3.7.1 entwickelten Filter wurde eine Möglichkeit geschaffen die Position mit dem Ultraschallbild zu verknüpfen. An sich stellt das MITK automatisch ein Bild in der Workbench dar. Dieses muss hierzu als *mitk::DataNode* der *mitk::DataStorage*-Klasse übergeben werden. Das Bild wird dadurch aber noch nicht richtig dargestellt. Abhängig von der Orientierung und Translation des Bildes kann es sein, dass man dieses gar nicht erkennt.
Liegt das Bild innerhalb der aktuellen Ansicht, werden die Schnittkanten mit den 2D Ansichten des MITK angezeigt. Über eine reinit-Funktion auf das Bild kann eine saubere Darstellung erfolgen. Dabei wird auf einer der 2D Ansichten das Ultraschallbild dargestellt. Auch in der 3D Ansicht wird das Bild korrekt dargestellt. Verändern sich jedoch die Koordinaten oder die Orientierung muss eine erneute Initialisierung auf das Bild erfolgen. Um das nicht von Hand machen zu müssen wurde eine Methode geschrieben, die abhängig von der eingestellten Frequenz eine Neuinitialisierung auf das Ultraschallbild ausführt. Hierbei wird zur Orientierung der MITK-eigenen Schichten (Bounding Box) die Orientierung und Translation des Ultraschallbildes verwendet. Da hier auch andere Bilder betroffen sind, sollte man diese Art der Darstellung nur anschalten, wenn man nicht auf anderen Daten arbeiten möchte. Daher kann im eigenen Plugin diese Darstellung über eine Checkbox an und ausgeschaltet werden.

### 3.7.3 Test der Darstellung

Zur Kontrolle der Darstellung wurde das Ultraschallbild mit dem Roboter-Tracking-System verknüpft. Anstelle von Werten des Roboters wurden Dummy-Werte verwendet. Diese simulierten das Drehen, Schwenken und Bewegen des Ultraschallkopfes. Zusätzlich wurde ein Punkt dargestellt, der die Position des Tracking-Systems zeigt. Es wurde geschaut ob sich dieser Punkt in der Mitte des Bildes am oberen Rand befindet. Bei einer Drehung um die z-Achse des Ultraschallgerätes konnte so kontrolliert werden ob sich das Ultraschallbild entsprechend um diesen Punkt dreht. Drehungen um die anderen Achsen wurden ebenfalls so getestet.

### 3.7.4 Registrierung und Darstellung mit Planungsdaten

Die Planung der Bestrahlung erfolgt auf MRT- und CT-Daten. Für eine spätere Anwendung ist es daher von zentraler Bedeutung die Ultraschallbilder mit den CT-Bildern zu registrieren. Weiter kann über ein registriertes Phantom die Kalibrierung des kombinierten Ultraschallbildes überprüft werden. Diese Überprüfung erfolgt über das Ultraschallphantom. Dieses wurde hierzu mit Markern ausgestattet und sowohl im MRT als auch CT eingescannt. Da die Bilder jedoch zu wenig Signal beinhalteten wurden die inneren Strukturen künstlich in das CT-Bild hineingesetzt. Hierfür wurden Zylinder mit entsprechenden Werten im Datensatz eingefügt. Die Positionierung erfolgte durch die Darstellung der inneren Strukturen auf der Rückseite des Phantoms. Es wurden dabei nicht alle Strukturen im CT-Bild entsprechend erzeugt. Ich möchte an dieser Stelle ausdrücklich Herrn Bendl für seine Unterstützung danken.
Über das Fiducial-Registration-Plugin lassen sich Marker und Bildpunkte in ein Koordinatensystem bringen. Das registrierte Bild wird dabei in das Koordinatensystem des optischen Tracking-Systems gebracht. Durch den entwickelten Filter wird das Ultraschallbild seinerseits entsprechend der Koordinaten gesetzt. Über die 2D Darstellung (3.7.2) erfolgt bereits eine automatisierte Darstellung des Ultraschallbildes. Das CT-Bild lässt sich durch einfache Transparenz mit dem Ultraschallbild überlagern. Durch die ständige Neuinitialisierung des Ultraschallbildes wird automatisch auch die dazugehörige CT-Ebene dargestellt. Ein Ergebnisbild ist in 3.8 dargestellt. Weitere Bilder sind im Ergebnis Kapitel 4.4.4. In der Abbildung ist zu erkennen, dass die inneren Strukturen im Ultraschallbild kleiner dargestellt werden und das Ultraschallbild leicht schräg zum CT-Bild versetzt ist. Die Größe der Strukturen könnte aber auch durch die nachträgliche Einfügung entstanden sein. Es konnte aber gezeigt werden, dass die Überlagerung an sich möglich ist.

## 3.8 3D Ultraschall

Für die spätere Verwendung ist ein 3D Ultraschall hilfreich. Mit ihm kann ein Bereich einmalig abgetastet werden, um z.B. eine ideale Schallebene zu finden. Da für

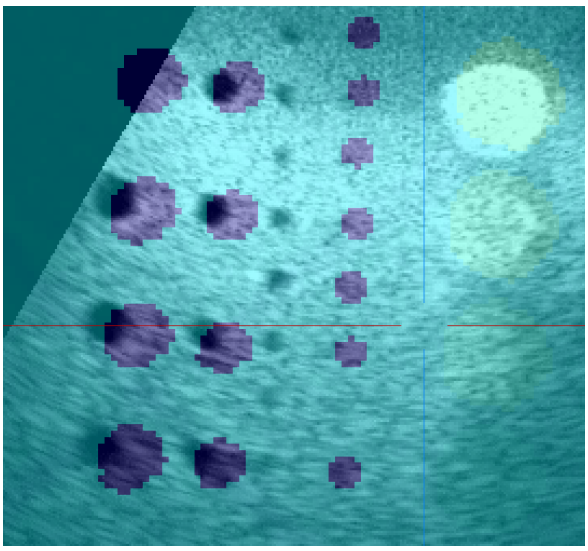


Abbildung 3.8: Überlagerte Darstellung des Registrierten Ultraschallbildes mit CT-Daten

die Versuche kein 3D Schallkopf zur Verfügung stand und dieser für die spätere Anwendung nicht zwingend notwendig ist, wurde eine Möglichkeit geschaffen aus den 2D Ultraschallbildern ein 3D Volumen zu rekonstruieren. Hierfür gab es bereits einige Vorarbeiten [12]. Diese waren durch schlechte Performance aber nicht verwendbar. Der generelle Ansatz, dass alle Bilder Voxel für Voxel in ein 3D Volumen übertragen werden, wurde übernommen. Die Art der Implementierung aber grundsätzlich verändert. Da der Ultraschallkopf erwartungsgemäß seine Ausgangslage nicht immens verändert, wurde entschieden das erste Bild für das Erstellen des 3D Würfels zu verwenden. Die Orientierung des Würfels ist dabei die Gleiche wie beim ersten Bild. Der Würfel wird dann so versetzt, dass das erste Bild in der Mitte des Würfels liegt. In Gegenrichtung des Schallkopfes werden 2 cm zusätzlich gegeben (siehe Abbildung 3.9). Das bestimmen des Würfels zu Beginn der Akquisition hat den Vorteil, dass die Bilder parallel zur Aufnahme in den Würfel eingetragen werden können.

Für die eigentliche Aufnahme wurde ein *QTimer* erstellt. Dieser triggert die Aufnahme und speichert das Bild in einem Array ab. Die Aufnahmefrequenz ließe sich theoretisch einstellen, ideal ist jedoch 30 Hz. Dadurch ist die Bildwiederholrate des Ultraschallbildes und die Speicherrate identisch. Als weitere Variable lässt sich die Anzahl der aufzunehmenden Bilder einstellen. Wichtig ist hier, dass die Aufnahmefrequenz und die Bildanzahl die Aufnahmedauer ergeben. Für ein 3D Ultraschallbild empfiehlt sich eine Aufnahmezeit die es ermöglicht dem Probanden während der Aufnahme die Luft anzuhalten. Daraus ergibt sich eine mögliche Aufnahmezeit von 6 s bis maximal 1 min. Das entspricht 180 bis 1800 Bildern.
Ein anderer Ansatz wäre es, die Bilder parallel darstellen zu lassen, bis der gewünsch-

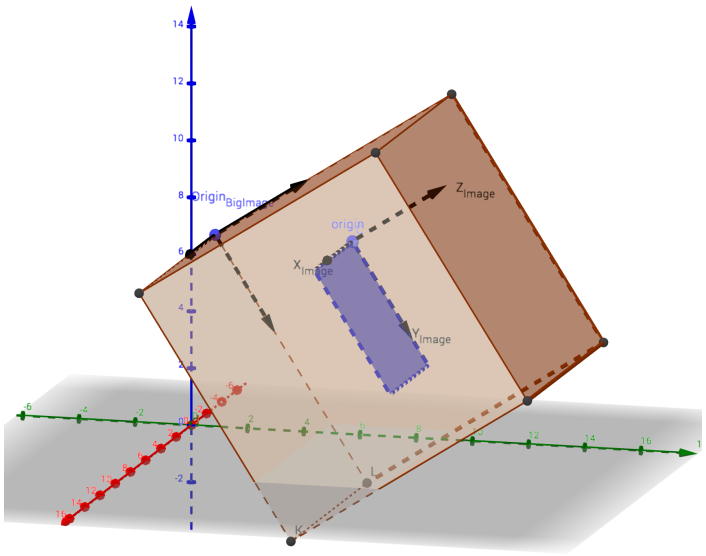



Abbildung 3.9: Verfahren zur Berechnung des 3D Würfels. Anhand des ersten Bildes werden die Randpunkte bestimmt. Das Blaue Rechteck entspricht hierbei dem Ultraschallbild im entsprechenden Weltkoordinatensystem.

te Bereich akquiriert ist und die Aufnahmezeit variabel zu lassen. Dieser Ansatz hat seine Vorzüge bei einem Freihandultraschall. Da das Ultraschallgerät sich jedoch am Roboter befindet ist die Annahme, dass die Bewegung exakt gesteuert werden kann. Somit sollten keine Bereiche ausgelassen werden. Zusätzlich wurde später versucht die Bilder parallel darstellen zu können, was nicht funktionierte.
Der verwendete QTimer kann gestoppt werden, wenn die gewünschte Anzahl an Bildern aufgenommen wurde. Die gespeicherten Bilder müssen nun in den erstellten Würfel geladen werden. Hierzu wurde über die Bilder iteriert und für jedes Voxel in jedem Bild die Weltkoordinate bestimmt. Anhand dieser wurde im Würfel die passende Indexkoordinate bestimmt und anschließend der Voxel des Würfels entsprechend gesetzt. Abschließend kann das Bild als *mitk::DataNode* in der Workbench betrachtet werden. Aus Usability Sicht wurde noch ein Fortschrittsbalken verwendet um dem Nutzer die Berechnungszeit und Akquisitionszeit kenntlich zu machen

### 3.8.1 Testen des 3D Ultraschall

Die im vorhergehenden Abschnitt entwickelte Funktion erstellt ein 3D Ultraschallbild. Es wurde überprüft ob diese den Anforderungen (3.2.2) entsprechend implementiert war. Damit das 3D Ultraschallbild später einen klinischen Nutzen haben kann, muss die Erstellung schnell gehen. Es wurde daher die Berechnungsdauer des 3D Ultraschallbildes überprüft.

- **Setzen der Positionsinformation:** ca. 3 ms. Dazu gehört das Umwandeln der Trackingdaten und das Speichern der Position in der Bild-Klasse.
- **Setzen der Bildinformation im Würfel:** ca. 70 ms. Dazu gehört die Übertragung jedes Voxels des Ausgangsbildes in den Würfel. Mit Umwandlung der Koordinaten.
- **Zeit zwischen den Aufnahmen:** 100 ms. Zeitabstände bis das nächste Bild gesetzt wird.

Aus den Messungen geht hervor, dass nicht alles funktionierte wie es sollte. Die Zeit zwischen den Aufnahmen sollte 30 ms betragen und war tatsächlich bei 100 ms teilweise sogar höher, teilweise immens niedriger. Eine Fehlersuche brachte als Ergebnis: Das Problem liegt an dem parallelen Anzeigen von Tracking-Geräten und Ultraschallbild in der Workbench. Über das Ultraschall-Plugin lässt sich die aktuelle Bildwiederholrate des Gerätes anzeigen. Diese liegt bei aktivierter Anzeige von Tracking-Werkzeugen bei teilweise unter 7 Hz. Durch die niedrigere Rate werden somit alle anschließenden Filteroperationen und der Zugriff aus dem Plugin heraus gebremst. Dieses Problem lässt sich lösen indem die Visualisierung der Tracking-Werkzeuge bei Erstellung des 3D Würfels deaktiviert wird.
Ein weiteres Problem das durch die Tests aufgedeckt wurde war, dass teilweise ein Bild nicht dargestellt werden konnte und das MITK sogar ganz abstürzte. Dieser Fehler trat auf, wenn die Bildebene in einem oder mehreren Winkeln im Bezug auf die MITK Standardebenen um die 45° liegt. Dann kann der Standard Renderer das Bild nicht initialisieren und die Ebenen auf dieses Bild anpassen. Zur Lösung des bekannten Problems wurde am *MITKCore*-Modul der Renderer entsprechend modifiziert[3].

### 3.8.2 Parallelisierung

Wie die Zeitmessungen zeigten, war die Berechnung des Ultraschallbildes nicht effizient. Durch die implementierte Funktion wurde viel Rechenzeit durch Warten verschenkt. Z.B. dauert das Setzen der Bildposition 3 ms und das nächste Bild wird danach in 27 ms akquiriert. Diese Zeit wird sonst nicht verwendet. Ein geeignetes Mittel um die Berechnungszeit des 3D Würfels zu beschleunigen ist somit die Parallelisierung von Akquisition und Setzen des Bildes in den 3D Würfel. Es wurde daher ein Thread definiert, der die Berechnung des Würfels und das Setzen der Bilder übernimmt. Der Thread muss dabei sicherstellen, dass er alle Bilder verwendet. Eine einfache Iteration über die bestehenden Bilder war damit nicht mehr möglich. Stattdessen durchläuft der Thread das Bilderarray und setzt das Bild an die entsprechende Stelle. Sollten alle Bilder abgearbeitet sein, wird überprüft ob alle Bilder bereits eingefügt worden sind. Ist das nicht der Fall wartet der Thread und setzt erst dann die Bilder ein. Dieses zusätzliche Konstrukt ist notwendig, sollte das Setzen des Bildes schneller gehen als die

[3] https://github.com/MITK/MITK/pull/203/

Bildaufnahme z.B. bei niedrig eingestellter Bildfrequenz. Zuerst wird wie im oberen Ansatz anhand des ersten Bildes der Würfel bestimmt.

### 3.8.3 Beschneidung des Ultraschallbildes

Weitere Rechenpower wurde verschenkt, indem alle Bildpunkte abgetastet worden sind. Über das Ultraschall-Plugin kann das Bild vom Ultraschallgerät grob zugeschnitten werden. Es wurde dabei wie folgt zugeschnitten: oben 175 px, unten 150 px, rechts 275 px und links 200 px. Dies ist für den Standard Ultraschallkopfwinkel von 60° ideal. Diese Konfiguration wird daher beibehalten. Die nachfolgende Verbesserung funktioniert so auch für andere Winkel. Jedoch werden bei größeren Schallwinkeln die Ränder bereits durch den ersten Zuschnitt abgeschnitten. Es wurde ein Algorithmus entwickelt, der die Ränder des Ultraschallbildes abtastet und diese abspeichert. Durch die Symmetrie des Ultraschallbildes reicht es eine Seite abzutasten. Der Algorithmus prüft ob ein Randpixel erreicht ist, indem er prüft ob die darauffolgenden Pixel 0 sind. Er arbeitet sich dabei von der Mitte des Bildes zu einer Seite hin vor. In der nächsten Zeile des Bildes beginnt er dann nicht mehr in der Mitte sondern in etwa an der Stelle des vorherigen Randes. Sind die Ränder so abgespeichert, muss der Algorithmus zum Setzen des Bildes nun nicht mehr das gesamte Bild ablaufen, sondern nur noch das eigentliche Ultraschallbild.
Für den vorher definierten Standardzuschnitt des Bildes ergibt sich dafür folgende Vereinfachung:

| | 30° | 45° | 60° | 75° | 85° |
|---|---|---|---|---|---|
| Bildanteil | 0,33% | 0,44% | 0,59% | 0,70% | 0,75% |
| Schwarzanteil | 0,67% | 0,56% | 0,41% | 0,30% | 0,25% |

Für passendere Zuschnitte entsprechend der Winkel sind diese Werte anders. Würde die erste Beschneidung passender je Winkel eingestellt, würde der größte Winkel durch den Algorithmus am meisten profitieren.
Da dieser Ansatz rechenintensiv ist, wird er vor der eigentlichen Messung gemacht. Für die Standardeinstellung wurden die Ränder bereits eingespeichert und lassen sich über die Benutzerschnittstelle auswählen. Eine grafische Darstellung der Randpunkte ist in Abbildung 3.10 zu sehen. Dieser Ansatz bringt nur bei einem Curved Array Ultraschallkopf etwas. Bei einem Linear Array, ist das ausgegebene Bild bereits rechteckig und kann exakt zugeschnitten werden. Wichtiger Nebeneffekt bei diesem Verfahren ist, dass das Logo des Ultraschallgeräteherstellers herausgeschnitten wird und kein Artefakt bildet.

## 3.9 Automation der Ultraschall 3D Akquisition mittels Roboter

Da der Ultraschallkopf direkt an dem Roboter montiert ist, lässt sich ein 3D Ultraschallbild erstmal nur durch Verwendung des Joysticks erstellen. Durch eine neue Funktion in der Robotersteuerung kann aber eine automatische Funktion erstellt werden,

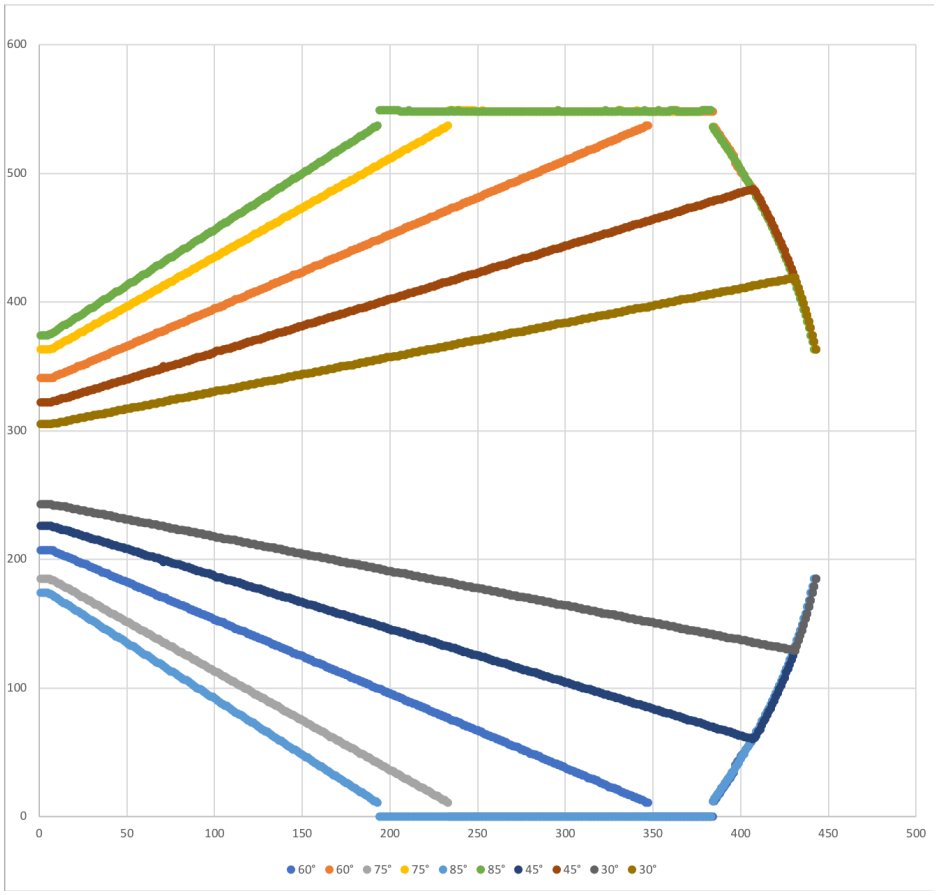



Abbildung 3.10: Ränder des Ultraschallbildes bei Standardzuschnitt.

welche die Aufnahme übernimmt. Dieser Abschnitt soll zeigen wie das automatisierte Scanverfahren entwickelt wurde.

Das in Abschnitt 3.7 entwickelte Verfahren zum Erstellen eines 3D Ultraschallbildes wurde mit dem erstellten Roboter-Tracking-Gerät getestet. Hierzu wurden Dummy-Werte eingespielt, die einen Rotations-Scan, Linearen-Scan, oder einen Schwenk-Scan simulierten. Diese Tests erlauben vor der eigentlichen Implementierung einen generellen Rückschluss über die bei der Erstellung notwendigen Parameter zu erhalten. Dazu gehört die Anzahl der Bilder sowie die Geschwindigkeit mit der der Roboter verfahren darf. Wenn der Vorschub des Ultraschallkopfes zu groß ist, entstehen Lücken im Bild. Für die spätere Verwendung ergibt sich dadurch eine maximale Scangeschwindigkeit. Diese hängt von der Größe des Ultraschallbildes ab. Bei der Standardeinstellung besitzt ein Ultraschallbild 0,33 mm pro Pixel. Bei einem linearen Vorschub des Ultraschallkopfes und 30 Hz Aufnahmefrequenz ergibt sich dadurch eine theoretische Geschwindigkeit von 30 mm/s. Neben der Implementierung des Scanverfahrens erfolg-

te daher ein Test zur Bestimmung der Scangeschwindigkeit.

### 3.9.1 Roboterapplikation

Auf Seiten des Roboters wurde die Kompensationsbewegung um eine Scanbewegung ergänzt. Hierbei wurde die Smart-Servo-Bewegung des Roboters verwendet. Diese erlaubt es dem Roboter zyklisch neue Positionen zu übermitteln. Parallel lässt sich die Atemkompensation über die Impedanz mit aufgeschalteter Kraft in z-Richtung des Werkzeugs aufrecht erhalten. Für eine Scanbewegung müssen die Randpunkte der Scanbewegung bestimmt werden. Der Roboter speichert sich hierzu die aktuelle Position ab. Im Falle eines **linearen Scanverfahrens** bestimmt er die Position vor und hinter der aktuellen Position. Die Geschwindigkeit dieser Bewegung wurde in Tests so ermittelt, dass ein lückenfreies 3D Volumen entsteht. Z.B. wurde ein Linear-Scan von 4 cm so realisiert, dass zuerst 2 cm in die eine Richtung verfahren wird, anschließend 4 cm entgegen dieser Richtung und zuletzt wieder 2 cm zum Ausgangspunkt zurück. Für die Bewegung wurde eine ideale Zeit von 12 s ermittelt. Die gefahrene Strecke entspricht in diesem Beispiel 8 cm. Daraus ergibt sich eine Scangeschwindigkeit von 6,6 mm/s. Die hierbei verwendete Smart-Servo-Bewegung überprüft dabei nicht ob das Ziel erreicht wurde, sondern nur ob die Koordinate der Scanachse erreicht wurde. Es kann nämlich sein, dass durch die Probandenoberfläche die Zwischenpunkte anatomisch gesehen nicht auf einer Ebene liegen.
Wie in Abschnitt 2.3.4 gezeigt gibt es neben dem Linear-Scan auch noch die Möglichkeit den Schallkopf zu schwenken oder zu drehen. Diese Verfahren wurden ebenfalls in der Robotersteuerung implementiert. Identisch zum Linear-Scan, wird hier die Smart-Servo-Bewegung mit Atemkompensation verwendet. Für einen Schwenk-Scan muss die Orientierung des Schallkopfes verändert werden. Hierbei kann ein Winkel definiert werden (Winkel um die x-Achse des Werkzeugs). Der erste Zielpunkt entspricht dann einer Umorientierung des Schallkopfes um die Hälfte des gesamten Winkels bei gleichbleibender Position. Der zweite Zielpunkt entspricht dann einer Umorientierung des Schallkopfes mit dem gesamten Winkel. Zuletzt wird die Orientierung wieder zur Ausgangslage zurückgesetzt. Der Rotations-Scan funktioniert etwas anders. Hier wird nur die Orientierung der Z-Achse um 180° verändert.

### 3.9.2 Test der Scanfunktionen

Die Scanfunktionen wurden iterativ entwickelt. Das bedeutet, Parameter für die Scanbewegung wie Länge, Winkel, Dauer und Geschwindigkeit wurden in Tests ermittelt. Hierzu wurden die Scanverfahren ausgeführt und die gewonnenen Bilder analysiert. Enthielt ein Bild zu viele Lücken wurde die Geschwindigkeit reduziert. Ergebnisbilder dieser Tests befinden sich in Anhang B ebenso der verwendete Versuchsaufbau 2.1.1. Im Falle des Linear-Scan wurde so die maximale Geschwindigkeit auf 0,66 mm/s gesetzt. Für den Rotations-Scan ergab sich, dass es nicht möglich war ein lückenloses

Bild zu erstellen, das um 180° gedreht ist. Weiter zeigte dieser Test, dass bei ungünstiger Ausgangslage des Roboters ein Sicherheitsrisiko besteht. In nicht allen Fällen kann der Ultraschallkopf um 180° gedreht werden. Im ungünstigsten Fall ist die Sicherheitsbewegung des Roboters durch die Achsstellung blockiert.

### 3.9.3 Einbindung ins MITK

Die erstellte Scanbewegung lässt sich über den *RobotManager* aufrufen. Dies ist die Verwaltungsklasse für den Roboter. Sie befindet sich im Roboter-Plugin und kann über die dazugehörige Benutzerschnittstelle erreicht werden. Die Steuerung für das 3D Ultraschall befindet sich jedoch außerhalb dieses Plugins im Ultraschall-Tracking-Plugin. Es wurde daher ein Service implementiert um die Funktionalität in beiden Plugins zu nutzen. Sobald der Roboter sich über OpenIGTLink verbindet, wird ein Service erzeugt. Der Service bietet jedoch keinen Direktzugriff auf den Roboter. Der Befehl wird erstmals im *RobotManager* überprüft. Sollte der Roboter sich nicht in einem passenden Modus befinden wird die Bewegung nicht ausgeführt. Der *RoboterService* ist als Singleton implementiert.

## 3.10 Test der gesamten Applikation

Zur Kontrolle der entwickelten Komponenten wurden Tests verwendet. Kleinere Tests befinden sich in den jeweiligen Abschnitten. In diesem Abschnitt werden die durchgeführten Tests des gesamten Systems erklärt. Es lassen sich hier zwei Haupttests unterscheiden. Zum Testen der Genauigkeit der korrekten Zuordnung des Ultraschallbildes und für die Registrierung mit einem CT-Bild wurde das Ultraschallphantom genutzt. Für den Nachweis einer klinischen Machbarkeit wurden Probandentests durchgeführt. Diese hatten als Ziel neben der Bildverarbeitung die Sicherheitskonzepte des Roboters zu Testen.

### 3.10.1 Test mit Ultraschallphantom

Anhand des Ultraschallphantoms sollten mehrere Dinge gezeigt werden. Dazu gehört:

- **Skalierung:** Es muss getestet werden ob die Skalierung des Ultraschallbildes im MITK richtig ist. Dabei muss auch geklärt werden ob diese im 2D Ultraschallbild und beim rekonstruierten 3D Ultraschallbild stimmt.
- **Orientierung:** Es muss überprüft werden ob die Orientierung des Ultraschallbildes richtig ist und z.B. nicht gespiegelt oder ähnliches.
- **Rekonstruktion des 3D Ultraschallbildes:** Es muss getestet werden ob das rekonstruierte Ultraschallbild die tatsächliche Struktur wiedergibt.

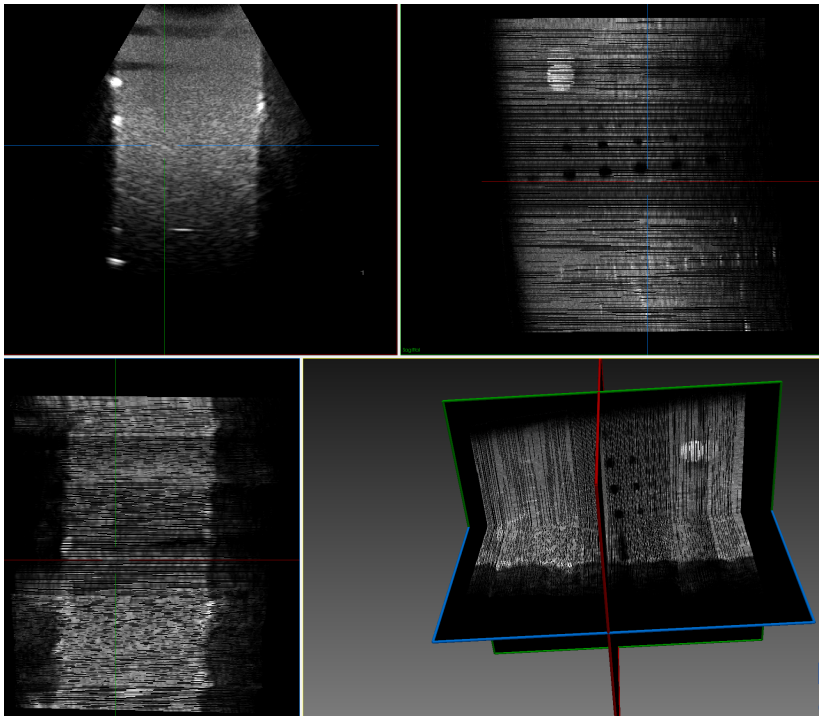


Abbildung 3.11: Ultraschallphantom orthogonal zur langen Seite gescannt. Die Lücken sind hierbei durch zu hohe Reibung entstanden. Der Roboter konnte dadurch nicht gleichmäßig über die Oberfläche gleiten. Zusätzlich war dieser zu schnell. Die größeren Strukturen wirken versetzt, bei der Aufnahme ist das Phantom an dieser Stelle verrutscht. Es zeigt dadurch welche Auswirkung Bewegung hat.

Zum Testen der vorangegangenen Punkte wurde das Ultraschallphantom linear abgescannt. Der Schallkopf wurde hierzu orthogonal zur langen Seite des Phantoms gehalten. Ergebnis war ein akquiriertes 3D Ultraschallbild des Phantoms. Zur Steuerung wurde der Roboter verwendet und über den Joystick manövriert. Das Ergebnisbild ist in Abbildung 3.11 dargestellt. An den großen Kreisen im oberen Bild lässt sich ein Versatz erkennen. Dieser entstand durch das Verrutschen des Phantoms. Da dieses nicht für die Vermessung benötigt wurde, wurde keine neue Aufnahme gemacht. Es zeigt aber welchen Einfluss Bewegung auf das 3D Ultraschallbild hat.
Das Phantom besitzt eine schematische Darstellung der Strukturen. Anhand dieser wurde überprüft ob die Strukturen im 3D Bild den Strukturen des Phantoms entsprechen. Somit konnte gezeigt werden, dass das Bild richtig skaliert ist und die Einzelbilder des 3D Bildes richtig zueinander orientiert sind.
Der Test zur Registrierung mit einem CT-Bild wird in Abschnitt 3.7.4 erklärt.

### 3.10.2 Test mit Probanden

Neben dem Nachweis der Richtigkeit der entwickelten Verfahren spielt die klinische Praktikabilität eine Rolle. In Probandentests wurde daher überprüft ob sich die entwickelten Funktionen in einen klinischen Workflow einbinden lassen. Es wurde hierfür ein kleines Szenario entwickelt:

1. Der Roboter befindet sich in der Startposition und alle Komponenten sind bereits initialisiert.
2. Der Proband legt sich auf die Liege.
3. An dieser Stelle kann eine Registrierung mit dem optischen System erfolgen. MRT-Daten standen zur Verfügung, jedoch fehlten passende Marker. Die Registrierung war dadurch zu ungenau für eine weitere Verwendung.
4. Der Roboter wird an der gewünschten Stelle in etwa platziert.
5. Es wird ein 3D Ultraschall gefahren und die ideale Schallebene gesucht.
6. Über den Joystick wird der Schallkopf entsprechend umorientiert.
7. Die Schallposition wird gehalten und das Ultraschallbild in 3D dargestellt. Ggf. mit Überlagerung der MRT-Daten.
8. Nach 5 Minuten wird der Versuch beendet und der Roboter an die Ausgangsposition verfahren.

Einzelne Punkte der oberen Reihenfolge wurden wiederholt durchgeführt. Z.B das Aufnehmen eines 3D Ultraschallbildes. Dieser Test wurde mehrfach wiederholt. Während des Versuchs war der Proband selbst am Notausknopf sowie ein Beobachter. Die Sicherheitssteuerung wurde ebenfalls geprüft. Bei zu starken Bewegungen wurde die 3D Bildgebung wie definiert abgebrochen und die Ausweichbewegung durchgeführt. Der Test konnte Nachweisen, dass sich der angegebene Ablauf mit den erstellten Plugins umsetzen lässt und dieser wiederholt funktioniert. Weiter konnte gezeigt werden dass verschiedene Positionen am Probanden für die neuen Funktionen verwendet werden können.
Eine zweiter Probandentest beschäftigte sich mit der 3D Rekonstruktion. In diesem sollten die Parameter für die automatisierten Scanverfahren ermittelt werden. Es wurden hierfür verschiedene Scanfunktionen getestet. Die Aufnahmezeit reichte von 6 s bis 1 min. Diese Tests wurden bereits in Abschnitt 3.9.2 beschrieben. Ergebnisbilder hierzu befinden sich in Anhang B. Neben den Parametern wurde zusätzlich überprüft in wie weit sich Bewegungen auf die 3D Bildgebung auswirken. Der Proband bewegte sich hierzu, z.B. wurde stärkeres ein- und ausatmen, Zuckungen, als auch noch gröbere Bewegungen simuliert. Ergebnisbilder dieses Tests befinden sich auch in Anhang B.5.

# 4 Ergebnisse

Ziel dieser Arbeit war die Konzeption und Entwicklung einer robotergestützten und ultraschallbasierten Lokalisationskontrolle. Die Hauptaufgabe dabei war die Referenzierung des Ultraschallbildes in einem entsprechenden Koordinatensystem. Erst durch eine solche Zuordnung der Bildpositionen lassen sich Strukturen im Bild eindeutig zuordnen. Ultraschall benötigt eine kontinuierliche ankopplung an die Patientenoberfläche. Um keine zusätzliche Person durch die Strahlenbelastung zu schädigen übernimmt diese Aufgabe ein Leichtbauroboter. Neben der Bildverarbeitung spielte somit die Integration eines Roboters als Halter des Ultraschallkopfes eine wichtige Rolle.
Ergebnis der Arbeit bildet eine auf MITK basierte Applikation. Diese erweitert MITK um neue Funktionen für die robotergestützte und ultraschallbasierte Lokalisationskontrolle. Dazu gehören Bildverarbeitungsfunktionen zum Verknüpfen von Tracking-Daten mit Ultraschallbildern, deren Darstellung, als auch Methoden zur Rekonstruktion eines 3D Ultraschallbildes aus 2D Ultraschallbildern. Weiter gehört zu den Funktionen eine Integration des Roboters in das MITK. Dieser kann nun aus anderen Plugins heraus verwendet werden und bietet neue Funktionen wie z.B. zum Erstellen eines automatisierten 3D Ultraschallbildes. Zusätzlich kann der Roboter als Tracking-System verwendet werden. Die neuen Funktionen wurden in Module und Plugins unterteilt und lassen sich dadurch unabhängig voneinander nutzen. Bereits vorhandene Funktionen des MITK wurden erfolgreich in den neuen Plugins genutzt. Die Applikation wurde getestet und erfüllt alle in 1.2 definierten Ziele. Zu den Ergebnissen gehört neben der Applikation die Erstellung und Kalibrierung eines Tracking-Werkzeugs, welches den Ultraschallkopf hält. Abschließend gehören die Erkenntnisse aus der Methodik (Kapitel 3) sowie diese selbst zu den Ergebnissen dieser Arbeit.
Dieses Kapitel baut sich wie folgt auf. Zuerst wird ein Überblick über das Gesamtkonzept gegeben. In diesem wird die Anwendung mit ihren Funktionen beschrieben. Zusätzlich zeigt dieser Abschnitt die Ergebnisse der Kalibrierung. Anschließend werden die in der Applikation erstellten Module und Plugins vorgestellt.

## 4.1 Gesamtkonzept

Für die Realisierung einer robotergestützten und ultraschallbasierten Lokalisationskontrolle wurde ein geeignetes Konzept entwickelt (Abschnitt 3.3). Dieses Konzept bildet neben der Umsetzung eines der wichtigsten Ergebnisse dieser Arbeit. Es ermöglicht die Integration von Roboter, Ultraschall, Joystick und Tracking-System in einer Anwendung. Es wurde dabei auf die universelle Verwendung der Komponenten geachtet. So lassen sich hardwareabhängige Einstellungen wie z.B. die Ultraschalltiefe in den dafür gegebenen Plugins einstellen. Das Konzept nutzt die bereits vorhandenen Module und Plugins und baut auf diesen auf. Es wurde somit erfolgreich an bestehende Funktionalität angeknüpft. Das Konzept erlaubt eine Weiterverwendung der Plugins. Nachfolgend werden die Hauptpunkte dieses Konzeptes gezeigt und die entsprechende Stelle an der die Ergebnisse präsentiert werden.

- **Roboter als Tracking-System:** Der Roboter lässt sich als Tracking-System verwenden. Die Verwaltung dieses Systems erfolgt über das Roboter-Plugin (4.3), die notwendigen Klassen enthält das Roboter-Modul (4.2). Die Beschreibung des Roboter-Tracking-Systems steht in Abschnitt 4.2.2.
- **Kombination von Tracking-Systemen:** Es ist möglich 2 Tracking-Systeme miteinander zu kombinieren, wenn die Basis des Werkzeugs identisch ist. Der hierbei erstellte Filter wird in Abschnitt 4.2.3 vorgestellt.
- **Kombination von Ultraschallbild und Tracking-Daten:** Ein Ultraschallbild kann durch einen erstellten Filter mit einem Tracking-Datensatz kombiniert werden. Dieser Filter wird in Abschnitt 4.2.4 gezeigt.
- **Darstellung des Ultraschallbildes in 3D und rekonstruierter 3D Ultraschall:** Das Ultraschllbild kann mit seiner Position dargestellt werden. Die Verwaltung der Anzeige erfolgt aus dem Ultraschall-Tracking-Plugin heraus (Abschnitt 4.4). Ebenso die Rekonstruktion des 3D Ultraschallbildes. Zusätzlich ermöglicht die Darstellung die Überlagerung eines Planungsbildes.
- **Roboterintegration:** Die bestehende Roboterintegration konnte erfolgreich umgewandelt werden. Der Roboter lässt sich jetzt auch aus anderen Plugins heraus verwenden. Es wurden automatisierte Scanfunktionen erstellt (4.5). Die Steuerung des Roboters erfolgt über das Roboter-Plugin (4.3).

Die Applikation beinhaltet 2 Module und 3 Plugins. Die Umstellung des ehemaligen einzelnen Plugins war erfolgreich. Die Funktionalität ist klar getrennt. Es gibt ein Roboter-Modul (4.2), dieses beinhaltet alle Funktionen die allgemein und pluginübergreifend verfügbar sein sollen. Dazu gehört z.B. die *RoboterService*-Klasse oder die erstellten Filter. Ein weiteres Modul ergab sich durch die Umstellung der Ausgangslage. Zur Steuerung des Roboters gibt es ein eigenes Plugin (4.3). Dieses beinhaltet die vorher erstellten Funktionen und neue Funktionen zum Testen der automatisierten 3D Ultraschallbildakquisition. Das Ultraschall-Tracking-Plugin (4.4) beinhaltet die Benutzeroberfläche zur Darstellung eines Ultraschallbildes mit Positionsinformation und zum Aufrufen von Funktionen für das Erstellen eines 3D Ultraschallbildes. Durch die Umstellung der Integration des Joysticks erhält dieser auch ein eigenes Plugin (4.7). Die erstellten Plugins in der Applikation werden in Abbildung 4.1 im Kontext des eigentlichen MITK dargestellt. Bei dem gezeigten Bild handelt es sich um ein mit den erstellten Plugins erzeugtes 3D Ultraschallbild von der Leber.

### 4.1.1 Kalibrierung

Zu den Ergebnissen gehört neben den implementierten Klassen auch die Erstellung eines Tracking-Werkzeugs. Für das optische Tracking-System wurde hierzu eine .rom-Datei erstellt (3.6.1). Die Position der Marker im Bezug zur neu definierten Basis lässt

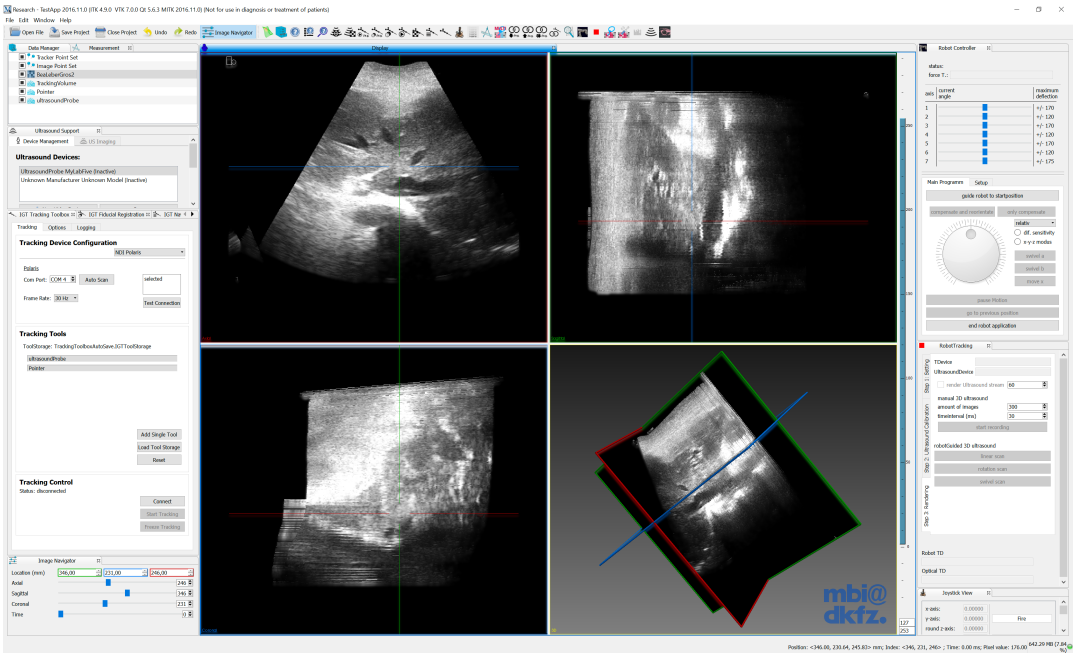


Abbildung 4.1: Das Bild zeigt die Workbench des MITK. Links befinden sind die verwendeten Plugins, rechts alle neu erstellten. Dargestellt ist ein aufgenommenes 3D Ultraschallbild.

sich hierbei Tabelle 4.1a entnehmen. Für den Roboter ist die Transformation im Bezug auf den Roboterflange in Tabelle 4.1b gegeben (3.6.2). Der FRE des optischen Systems entspricht 1,27 mm. Der Berechnungsfehler beim Erstellen des Roboter-Tracking-Systems betrug 0,99 mm. Es wurde hierbei versucht beide Tracking-Systeme auf die selbe Basis zu eichen.

Tabelle 4.1: Transformationen für den Ultraschallkopf

(a) Transformation für die optischen Marker

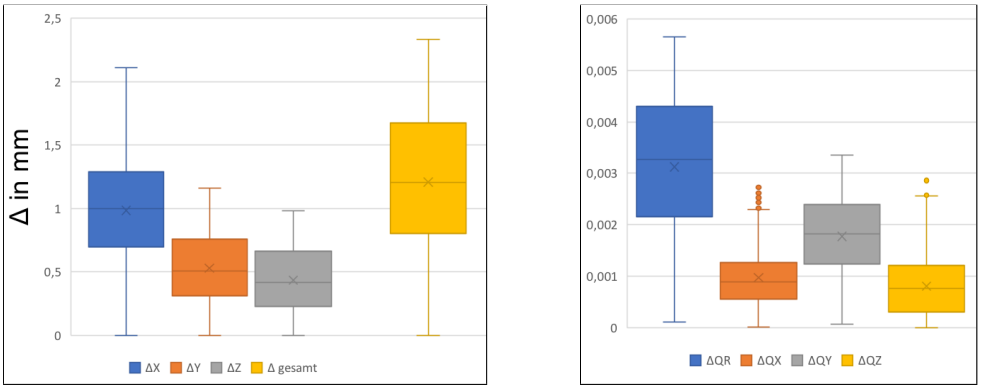
| Marker | x | y | z |
|---|---|---|---|
| A | -30,7 | -24,5 | -131,23 |
| B | -30,7 | 0 | -95,95 |
| C | -30,7 | 0 | -156,86 |
| D | -30,87 | 39,48 | -120,16 |

(b) Transformation vom Roboterflange

| x | y | z | a | b | c |
|---|---|---|---|---|---|
| -79,8 | -2,7 | 114,7 | 0 | 0 | 0 |

Die tatsächliche Abweichung der Basen wurde in Abschnitt 3.6.4 getestet. Es wurde überprüft in wie weit sich anhand der Daten des einen Tracking-Systems, die Daten des anderen Systems berechnen lassen. Es ergibt sich eine Abweichung in der Translation um im Median 1 mm in x, 0,5 in y und 0,4 in z-Richtung (siehe Abbildung 4.2a).Die

Orientierung in Quaternion-Darstellung ist im Median um zwei Nachkommastellen identisch (siehe Abbildung 4.2b).




(a) Abweichung XYZ (b) Abweichung Quaternion

Abbildung 4.2: Abweichung der transformierten Werte zum original Tracking-Gerät

Das Ultraschallbild besitzt seinen Ursprung in der unteren linken Ecke. Es wird daher eine weitere Transformation benötigt um das Bild entsprechend mittig an der definierten Basis in der Ultraschallkopfspitze zu positionieren. Durch die Definition der Basis des Tracking-Werkzeugs entspricht die Transformation in homogenen Koordinaten:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -UDimension * Scaling * 0.5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Diese dient als Kalibrierungsmatrix und wird mit den Tracking-Daten verrechnet. Dadurch ist sichergestellt das unabhängig von der Größe des Ultraschallbildes und des Tracking-Werkzeugs das Bild richtig im Raum zugeordnet wird.

## 4.2 Roboter-Module

Das Roboter-Modul beinhaltet die wichtigsten Klassen für das Roboter-Plugin und das Ultraschall-Tracking-Plugin. Es ist vor allem wegen der *RoboterService*-Klasse notwendig, da der Roboter sonst nicht aus verschiedenen Plugins angesprochen werden könnte. Weiter enthält es die erstellten Filter. Eine genauere Übersicht der Klassen folgt im nächsten Abschnitt.

### 4.2.1 Aufbau

Nachfolgend wird der Aufbau des implementierten Moduls gezeigt. Die Funktion der erstellten Klasse steht jeweils dabei.

- **Modul DeviceRegistry**: Autoload-Modul zum Erstellen eines *RoboterService* zu Beginn der Anwendung.
  - **RobotActivator**: Activatorklasse registriert den *RoboterService*.
- **ChangeOrientationMessage**: Nachrichtenklasse mit Positionsdaten, z.B. zum Schwenken des Ultraschallkopfes.
- **mitkPoseMessage**: Nachrichtenklasse mit der Achsstellung des Roboters.
- **mitkRobotTrackerTypeInformation**: Basisklasse für die Definition von Tracking-Geräten.
- **mitkRobotTrackingDevice**: Implementierung des Roboter-Tracking-Geräts.
- **mitkRobotTrackingTool**: Werkzeug für das Roboter-Tracking-System.
- **RobotService**: Service für Funktionen des Roboters die in anderen Plugins genutzt werden können.
- **UltrasoundTrackingCombineFilter**: Filter um zwei Tracking-Geräte miteinander kombinieren zu können.
- **UltrasoundTrackingVisualizationFilter**: Filter zum Setzen der Positionsinformation in Bildern anhand von Tracking-Daten.

### 4.2.2 Roboter-Tracking-System

Der Roboter kann als normales Tracking-System im MITK verwendet werden. Er kann dabei die Koordinaten des Roboterflange oder einen erstellten Referenzpunkt des montierten Werkzeugs liefern. Das Zeitintervall zum übermitteln der Position ist einstellbar. Entsprechend der Ultraschallbildwiederholrate sendet der Roboter die Position der Ultraschallkopfspitze alle 30 Hz. Die Daten des Roboters sind bei bestehen der Verbindung immer valide. Die Steuerung des Tracking-Geräts erfolgt im *Roboter-Plugin*. Die Klassen des Tracking-Geräts sind jedoch in diesem Modul hinterlegt. Die Implementierung entspricht den Leitlinien für Tracking-Geräte im MITK. Es könnte daher auch aus dem IGT-Plugin heraus verwendet werden.

### 4.2.3 Tracking-Kombinations-Filter

Zum Verknüpfen der beiden Tracking-Systeme (Roboter und optisches) wurde ein Navigation-Data-to-Navigation-Data-Filter erstellt. Da für die Anwendung nur ein Werkzeug kombiniert werden muss benutzt der Filter zwei Eingänge. Der erste Eingang definiert das normal verwendete Tracking-System. Im Kontext der Arbeit wurde der Eingang hierzu mit dem optischen Tracking-System verbunden. An den zweiten Eingang wird das alternative Tracking-System angeschlossen. In diesem Fall das Roboter-Tracking-System. Zu Beginn kann anhand der Positionsinformation von beiden Geräten eine Transformationsmatrix bestimmt werden, welche die Tracking-Daten des zweiten Eingangs zu entsprechenden Tracking-Daten des ersten Eingangs umwandelt. Am Ausgang des Filters liegen dann generell die Daten des ersten Eingangs an. Sollte dieser keine validen Daten liefern, werden die transformierten Daten des zweiten Eingangs genutzt. Die Transformation des zweiten Eingangs erfolgt nur bei invaliden Daten des ersten Eingangs. Abbildung 4.3 zeigt den Filter schematisch. Der Quellcode der *GenerateData*-Methode befindet sich im Anhang C.2.

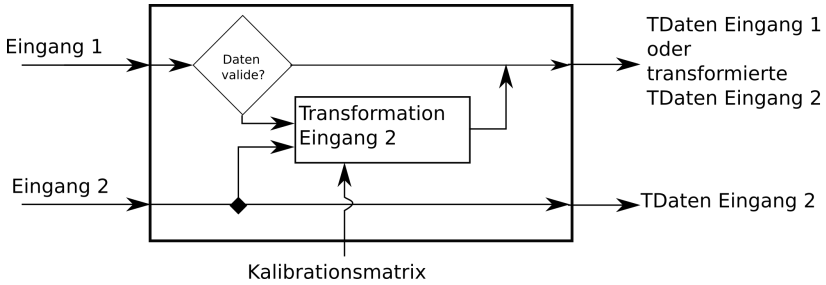



Abbildung 4.3: Schematische Darstellung des Tracking-Kombinations-Filters. Die Daten des ersten Ausgangs werden entweder durch den Eingang 1 gesetzt oder wenn dieser nicht valide ist durch Eingang 2. An Ausgang 2 liegen einfach nur die Daten des Eingang 2 an.

### 4.2.4 Ultraschall-3D-Filter

Der erstellte Ultraschall-3D-Filter ist eine Kernkomponente. Es handelt sich hierbei um einen klassischen Navigation-Data-to-Navigation-Data-Filter. Der Filter an sich leitet die Tracking-Daten der Eingänge dabei unverändert an die Ausgänge weiter. Dem Filter kann ein Ultraschallgerät bzw. eine Bildquelle übergeben werden. Für die Kalibrierung des Bildes muss zusätzlich eine Transformationsmatrix übergeben, sowie die Skalierung des Bildes (Spacing) gesetzt werden. Die Bildinformation wird dann entsprechend der Tracking-Daten richtig gesetzt. Das bedeutet, dass das Bild richtig skaliert, orientiert und entsprechend translatiert wird. Die Transformationsmatrix, das Spacing und das Ultraschallgerät müssen einmalig gesetzt werden. Die eigentlichen Eingänge des Filters lassen sich mit dem optischen Tracking-System, dem Roboter

oder dem erstellten Tracking-Filter verknüpfen. Liegen neue Tracking-Daten an reicht sie der Filter automatisch weiter und lädt sich parallel das nächste Bild von der Bildquelle. Dieses wird mit der Position verknüpft und kann über einen eigenen Ausgang an weitere Operatoren übergeben werden. Schematisch ist das ganze in Abbildung 4.4 festgehalten. Der Quellcode der *GenerateData*-Methode befindet sich im Anhang C.1.

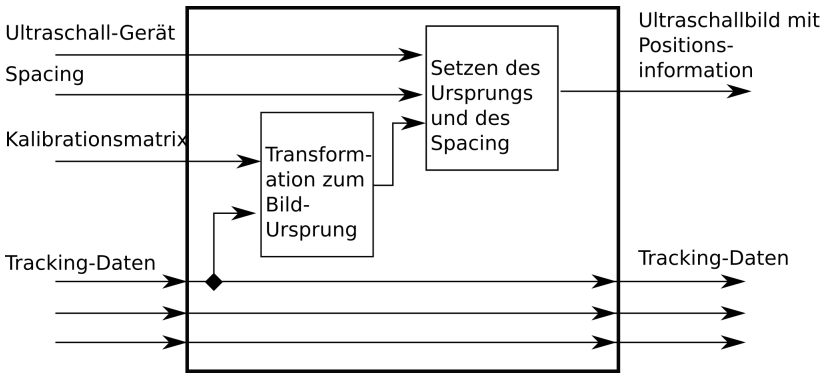



Abbildung 4.4: Schematische Darstellung des Ultraschall-3D-Filters. Die Eingänge Ultraschallgerät, Spacing und Kalibrationsmatrix werden nur einmal gesetzt.

### 4.2.5 Roboter Service

Um anderen Plugins einen Zugriff auf Funktionen des Roboters zu ermöglichen wurde in diesem Modul ein entsprechender Service implementiert. Dieser verwaltet ein Interface, in welchem die Funktionen definiert sind. Über das Roboter-Plugin kann somit das Interface implementiert werden und über andere Plugins die tatsächliche Funktion genutzt werden. Momentan ist nur eine Funktion erstellt. Dabei handelt es sich um den Befehl zur automatischen Aufnahme eines 3D Ultraschalls. Übergeben werden kann die Zeit bis zur Aufnahme und der Scan-Modus (schwenken, rotieren oder linear).

## 4.3 Roboter-Plugin

Das Roboter-Plugin entwickelte sich im wesentlichen aus dem vor der Masterarbeit vorhandenen Plugin. Da sich intern jedoch einiges geändert hat, gehört auch dieses Plugin zum Gesamtergebnis. Das Roboter-Plugin übernimmt die Steuerung des Roboters. Es bietet hierzu eine Benutzerschnittstelle an. Für diese wurde die Benutzerinteraktion überarbeitet und verbessert. Dabei wurden Fehler durch eine Nutzereingabenvalidierung behoben. Das bedeutet, dass ein Button nun deaktiviert wird, wenn er nicht verwendet werden kann. Außerdem wird die Oberfläche erst angepasst, wenn die Robotersteuerung ein Kommando tatsächlich umgesetzt hat oder wird geändert,

wenn z.B. eine durch den Roboter getriggerte Sicherheitsbewegung durchgeführt wurde. Zusätzlich wurden Fehlernachrichten und Rückmeldungen implementiert. Weiter lassen sich die erstellten automatisierten Scanfunktionen auswählen und damit auch ohne Bildgebung testen. Sobald der Roboter verbunden ist, wird der *RoboterService* initialisiert und der Roboter steht auch anderen Plugins zur Verfügung.

### 4.3.1 Aufbau

Das Plugin setzt sich aus folgenden Klassen und Submodulen zusammen:

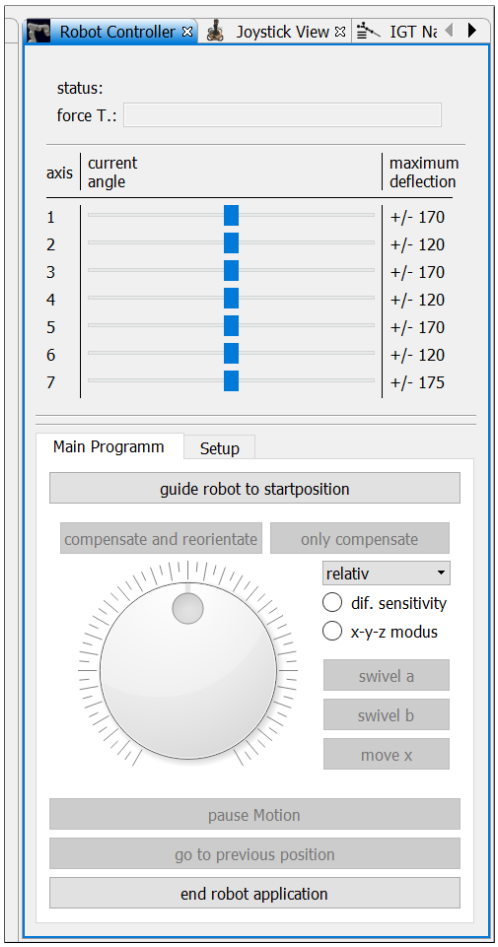
- **hhn_mirobotproject_robotcontrollerActivator**: Standard Activatorklasse für das Plugin.
- **RobotController**: Verwaltet die Benutzeroberfläche. Stellt den Status des Roboters dar und sperrt Funktionen auf der Benutzeroberfläche entsprechend.
- **RobotManager**: Repräsentiert den Roboter im MITK. Verwaltet dessen Funktionen und koordiniert die Zugriffe auf diesen.
- **RobotOpenIGTLinkServer**: OpenIGTLink-Server zum Verbinden mit dem Roboter. Notwendig, da eigene Nachrichtenklassen entwickelt wurden.

### 4.3.2 Erweiterungen Scanbewegung und Tracking-Gerät

Das Plugin enthält Funktionen für automatisierte Scanbewegungen und für das erstellte Roboter-Tracking-System. Es wurden drei verschiedene Scanverfahren hinterlegt. Linear-Scan, Schwenk-Scan und Rotations-Scan. Diese sind vordefiniert. Grundsätzlich erlaubt die erstellte Implementierung auch diese Bewegungen variabler zu gestalten, z.B. längere Strecke im Linear-Scan. Die dafür notwendigen Funktionen sind implementiert. Damit die Anwendung später nicht zu komplex wird, wurde auf diese Variabilität erstmal verzichtet. Zusätzlich lassen sich diese Scanfunktionen aus anderen Plugins heraus ausführen. Es wird dabei überprüft ob die Bewegung zulässig ist. Das Plugin verwaltet außerdem das neu erstellte Roboter-Tracking-System. Über die Benutzeroberfläche lässt sich dieses starten und beenden.

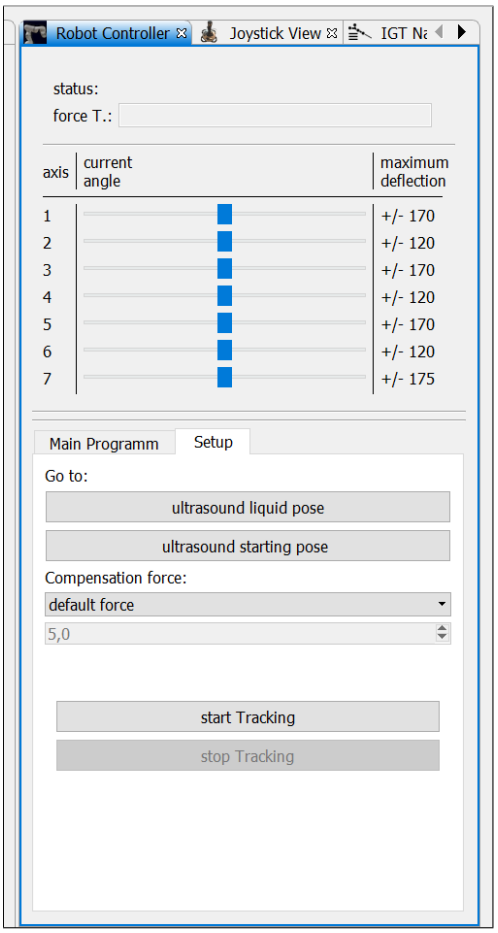
### 4.3.3 Benutzerschnittstelle

Die Benutzerschnittstelle zeigt in der oberen Hälfte den Status des Roboters. Dazu gehören der aktuell ausgeführte Befehl, der Druck auf das Werkzeug und die Achsstellung des Roboters. Die Achsstellung ist dabei farbcodiert und zeigt so, ob sich eine Achse zu sehr am Rand befindet. Diese Anzeige ist z.B. wichtig beim Platzieren des Ultraschallkopfes. Im Setup-Tab kann der Roboter zu gespeicherten und vordefinierten Positionen gefahren werden. Es kann weiter der Druck des Ultraschallkopfes angepasst werden. Neu dazugekommen ist der Button zum Starten und Beenden des




(a) Hauptfenster der Roboterapplikation

Hauptfenster der Roboterapplikation. Es bietet die Grundfunktionen um den Roboter von Hand zu führen, zwischen den Kompensationsmodi zu wechseln und die Orientierung und Translation über den Joystick zu ändern. Weiter erlaubt es die Achsstellung des Roboters über ein Rad zu ändern. Vordefinierte Scanfunktionen lassen sich zum Testen auch ohne Ultraschall aufrufen.




(b) Setup-Tab

Setup-Tab der Roboterapplikation. Dieser erlaubt es den Roboter in eine Startposition oder in eine Position zum Gel auftragen zu steuern. Es lässt sich der Anpressdruck des Ultraschallkopfes einstellen. Neu dazu gekommen ist außerdem das Starten und Beenden des Trackings.

Abbildung 4.5: Benutzerschnittstelle des Roboter-Plugins

Roboter-Tracking-Systems. Im Hauptprogramm-Tab kann der Roboter von Hand geführt werden. Zwischen einfacher Kompensation und Kompensation mit Joystick kann gewechselt werden. Der Joystick kann für Translation oder zum Schwenken des Ultraschallkopfes verwendet werden. Neu hinzugekommen sind die vordefinierten Scanfunktionen. Diese können per Knopfdruck entweder eine Schwenk-, Dreh- oder Linearbewegung des Roboters auslösen. Durch das verwendete Design könnten hier aber auch komplett neu definierte Bewegungen hinterlegt und ausgeführt werden. Die Achsstellung des Roboters lässt sich über das Rad anpassen. Ansonsten kann die Applikation wie vorher pausiert oder beendet werden.

## 4.4 Ultraschall-Tracking-Plugin

Die Kernaufgabe der Arbeit war das Verknüpfen von Bild und Positionsinformation. Die hierzu notwendigen Filter wurden bereits im Roboter-Modul erklärt. Diese Filter kommen aber nicht ohne eine Benutzerinteraktion aus. Das Ultraschall-Tracking-Plugin wurde daher entwickelt um die Hauptaufgabe der Arbeit zu realisieren. Es beinhaltet sowohl die Schritte zum Initialisieren und Einrichten der Filter, als auch die Funktionen zur Darstellung des gefilterten Ultraschallbildes. Weiter kann das Plugin über gespeicherte Ultraschallbilder ein 3D Ultraschallbild errechnen. Folgende Funktionen werden in dem Plugin umgesezt:

- Verbindung der Tracking-Geräte
- Setzen der Einstellungen des Ultraschallbildes
- Darstellung des Ultraschallbildes in 3D
- Aufnahme und Darstellung eines 3D Ultraschallbildes
- Automatisierte 3D Ultraschallbildakquisition

### 4.4.1 Aufbau

- **hhn_mi_robotproject_ultraschallTracking_Activator**: Standard Starterklasse.
- **UltraschallTracking**: Benutzerschnittstellen-Verwaltungs-Klasse.
- **UltraschallTrackingManager**: Threadklasse zum Aktualisieren der Tracking-Daten in der Benutzerschnittstelle. Beinhaltet eine weitere Threadklasse *Ultraschall3DWorkingThread*, welche die Erstellung und Parallelisierung für das 3D Ultraschallbild übernimmt.

### 4.4.2 Benutzeroberfläche

Es wurde eine Benutzeroberfläche geschaffen, die es dem Benutzer erlaubt anhand von simplen Schritten die benutzten Geräte zu verknüpfen und die erstellten Filter zu verwenden. Die Benutzeroberfläche teilt sich dafür in 3 Schritte ein:

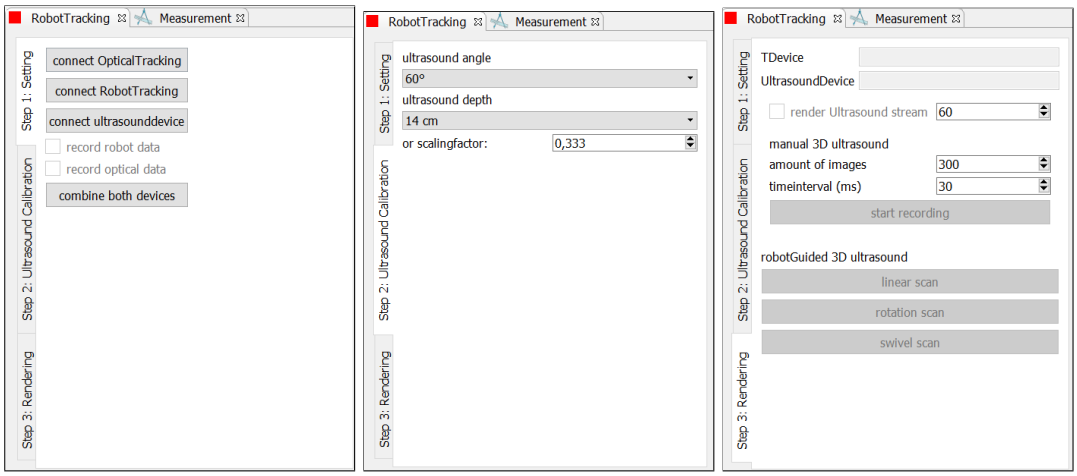
- **Step 1 Setting**: Im ersten Fenster (siehe Abb. 4.6a) befindet sich alles nötige um die verwendeten Geräte zu laden. Im ersten Schritt kann das Plugin so mit dem optischen Tracking-System, Roboter-Tracking-System und dem Ultraschallgerät verbunden werden. Bei den geladenen Tracking-Geräten kann ausgewählt werden ob deren Daten in eine Datei geschrieben werden sollen. Über den *combine both devices*-Button lassen sich die Tracking-Systeme verbinden und der Tracking-Kombinations-Filter wird anhand der momentanen Position des Ultraschallkopfes initialisiert.
- **Step 2 Ultraschall Kalibrierung**: Über den Kalibrierungs-Tab lassen sich die eingestellten Werte des Ultraschallgerätes übernehmen. Dazu gehört der eingestellte Winkel sowie der Skalierungsfaktor. Ersteres dient der effizienteren Berechnung des 3D Bildes, letzteres bestimmt die Skalierung des Ultraschallbildes und muss richtig gesetzt sein.
- **Step 3 Anzeigen**: Hier werden das verwendete Ultraschallgerät und das Tracking-Gerät angezeigt. Über eine Checkbox kann das Ultraschallbild in 3D angezeigt werden. Die Framerate lässt sich dabei einstellen. Für das 3D Ultraschall stehen mehrere Möglichkeiten zur Verfügung. Es kann ein komplett frei eingestelltes Ultraschallbild erstellt werden. Frequenz und Anzahl der Bilder ist einstellbar. Über 3 weitere Buttons lassen sich die automatisierten Scanverfahren des Roboters abrufen.

Im unteren Bereich werden parallel die Daten der Tracking-Geräte angezeigt.

### 4.4.3 Ultraschallbild in 3D

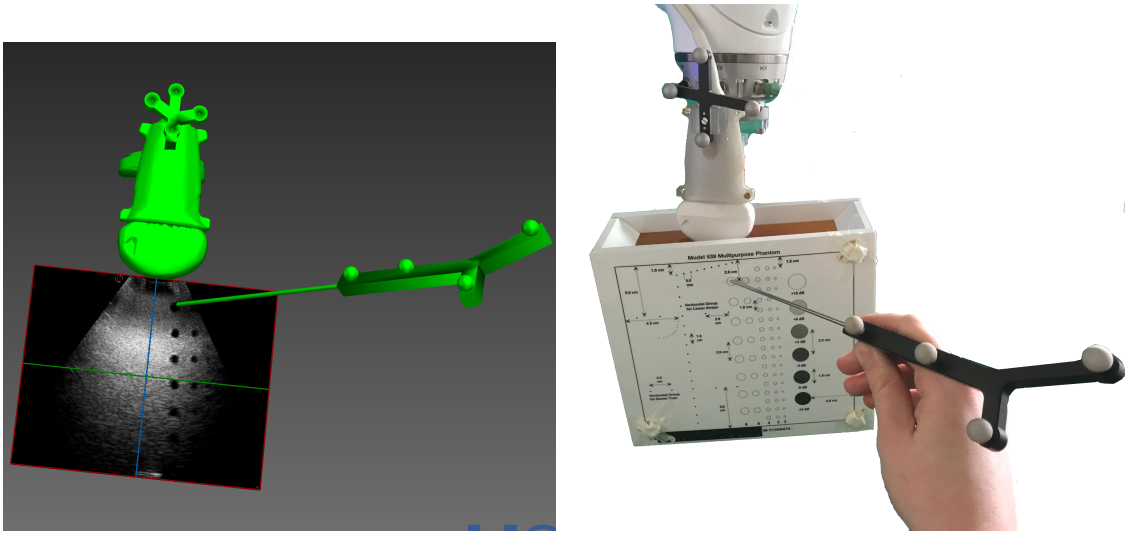
In Abbildung 4.7a ist das Ergebnis des Ultraschallbildes in 3D zu sehen. Man kann erkennen, dass das Ultraschallbild an dem Tracking-Werkzeug entsprechend dranhängt. Die Anzeige in der Workbench spiegelt exakt die in Abbildung 4.7b dargestellte reale Situation wieder. Das Bild zeigt, dass die implementierte Pipeline funktioniert und die Kalibrierung stimmig ist. Das Ultraschallbild kann mit der Positionsinformation im MITK dargestellt werden.

Es kann gezeigt werden, dass das Plugin es ermöglicht das Ultraschallbild in quasi Echtzeit in 3D darzustellen. Es verwaltet hierzu den Ultraschall-3D-Filter. Die Darstellungsfrequenz ist variabel und kann eingestellt werden. Sie kann jedoch nicht schneller sein als die Aufnahmefrequenz des Ultraschallgerätes. Zusätzlich getrackte Werkzeuge können in der Workbench parallel dargestellt werden. Es ist somit möglich anhand des




(a) Setting-Tab (b) Kalibrations-Tab (c) Anzeige-Tab

Abbildung 4.6: Benutzerschnittstelle des Ultraschall-Tracking-Plugins



(a) Ansicht Workbench (b) Versuchsaufbau in real

Abbildung 4.7: Ultraschallbild in 3D

Ultraschallbildes und den sich darin befindlichen Strukturen einen Rückschluss auf deren momentane Position zu ziehen. Wie Abbildung 4.8 zeigt, wird das Ultraschallbild sowohl in der 3D Ansicht, als auch in einer Ebenenansicht des MITK dargestellt. Weiter zeigt dieses Bild einen Test mit einem Probanden. Dieser konnte zeigen, dass sowohl die Ankopplung des Ultraschallkopfes durch den Roboter, als auch die Darstellung möglich ist. Anhand der Darstellung kann nun die Kompensationsbewegung indirekt dem Benutzer in der 3D Ansicht der Workbench sichtbar gemacht werden. Die Anzeige sowie das Bild werden automatisch aktualisiert und als Stream wiedergegeben. Dieser Stream kann zukünftig weiter verwendet werden.

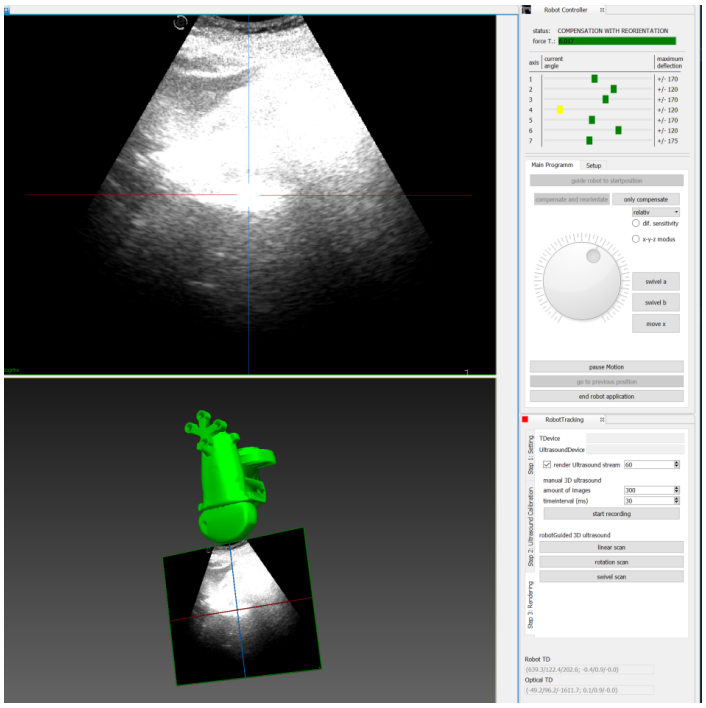
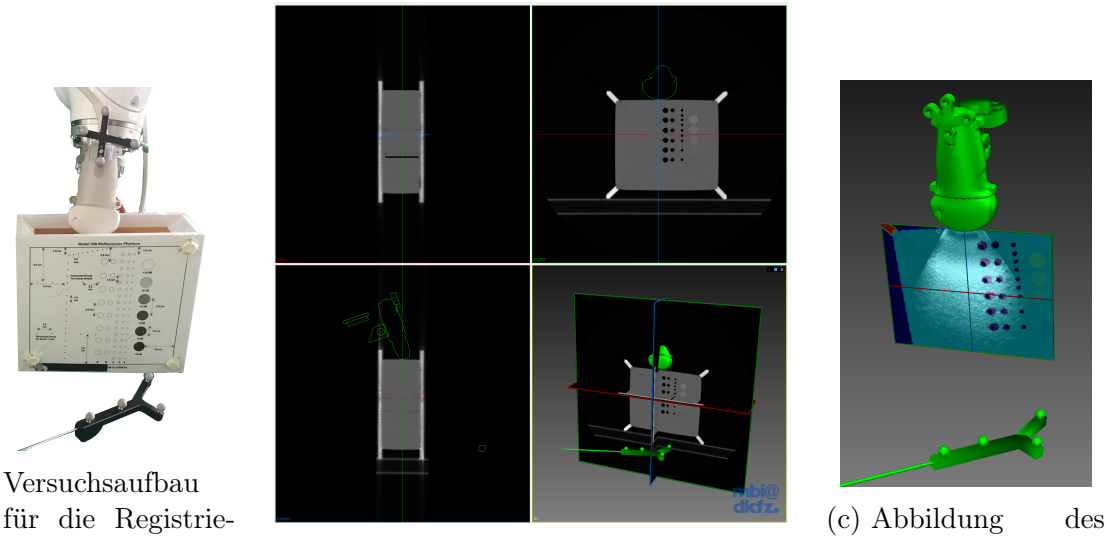



Abbildung 4.8: Workbench MITK mit 3D Ansicht des Ultraschallbildes

### 4.4.4 Registrierung Planungs-Bild

Eine Registrierung mit einem Planungs-Bild ist möglich. Im Folgenden ist dieses Planungsbild beispielhaft ein CT-Datensatz. Ein über das optische Tracking-System registrierter CT-Datensatz kann parallel zum Ultraschallbild dargestellt werden. Abbildung 4.9 zeigt den Ablauf einer solchen Registrierung. links ist der Versuchsaufbau dargestellt. In der Mitte das registrierte CT-Bild und rechts die überlagerte Darstellung des Ultraschallbildes mit der entsprechenden Schnittebene des CT-Bildes.

Bei dem verwendeten CT-Datensatz handelt es sich um einen realen Datensatz des Phantoms. Dieser beinhaltete jedoch keine Strukturen. Es wurden hierfür nachträglich



(a) Versuchsaufbau für die Registrierung mit dem CT-Bild

(b) Registrierter CT-Datensatz mit Tracking-Werkzeugen

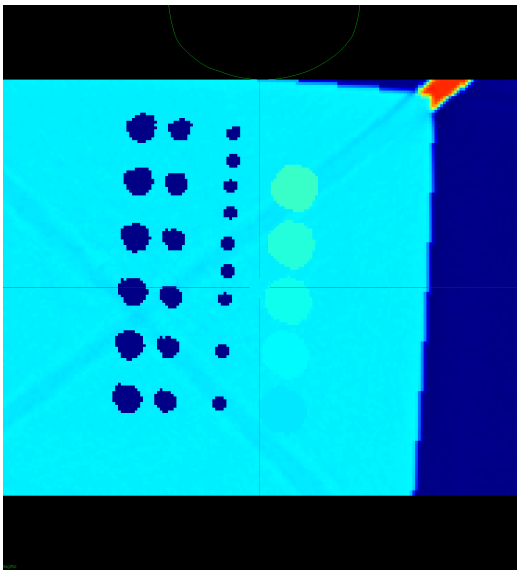
(c) Abbildung des Versuchsaufbaus in der Workbench

Abbildung 4.9: Darstellung des Registrierungsablaufs. (a) zeigt das reale Setup. (b) zeigt das Ergebnis der Registrierung und (c) zeigt die neue Überlagerung der beiden Modalitäten in Echtzeit und 3D.

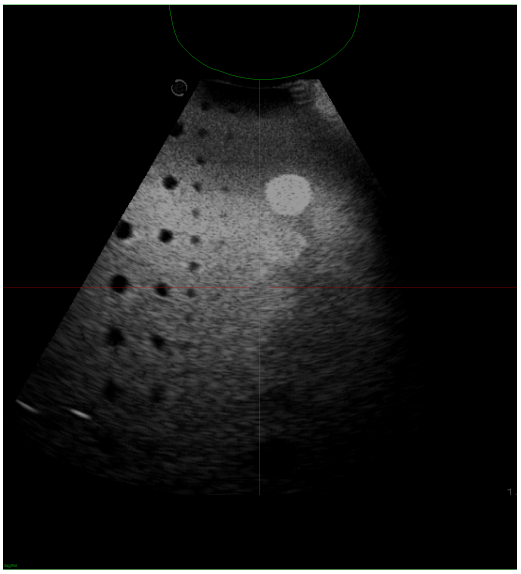
Zylinder manuell in den CT-Datensatz gesetzt. Hierfür wurden die Zylinder entsprechend der Abbildung 4.10c gesetzt. Es wurden nur ein paar der tatsächlich vorhandenen Strukturen verwendet. In der überlagerten Darstellung (Abbildung 4.10d) ist dies zu erkennen. Der FRE der Registrierung des Phantoms war bei 0,57 mm. Anhand der Bilder lässt sich erkennen, das die Strukturen noch nicht vollständig übereinander liegen. Daraus ergibt sich als Ergebnis, dass eine präzisere Kalibrierung in einem nächsten Schritt entwickelt werden muss.

### 4.4.5 3D Ultraschallbildrekonstruktion

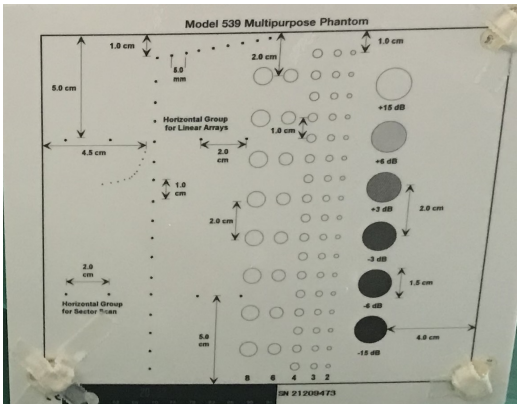
Damit bei einer Verwendung eines 2D Ultraschallkopfes eine Übersicht über das zu untersuchende Gebiet erstellt werden kann wurde eine 3D Ultraschallfunktion implementiert. Es lässt sich dabei einstellen wie viele Bilder für ein 3D Ultraschallbild verwendet werden sollen. Es können dabei zwischen 10 und 1800 Bilder verwendet werden. Die obere Schranke entspricht einer Aufnahmezeit von einer Minute bei 30 Hz Aufnahmefrequenz. Der Algorithmus erlaubt es noch viel mehr Bilder aufzunehmen. Die Grenze wurde hier willkürlich gesetzt und entspricht dabei der Zeit, in der ein trainierter Mensch die Luft anhalten kann. Neben der Anzahl kann auch die Aufnahmefrequenz eingestellt werden. Hierbei ist 30 Hz die obere Grenze. Dies entspricht der Aufnahmefrequenz des Ultraschallgerätes. Damit die Aufnahmefrequenz stimmt müssen alle momentan dargestellten Objekte im MITK deaktiviert werden. Das bezieht



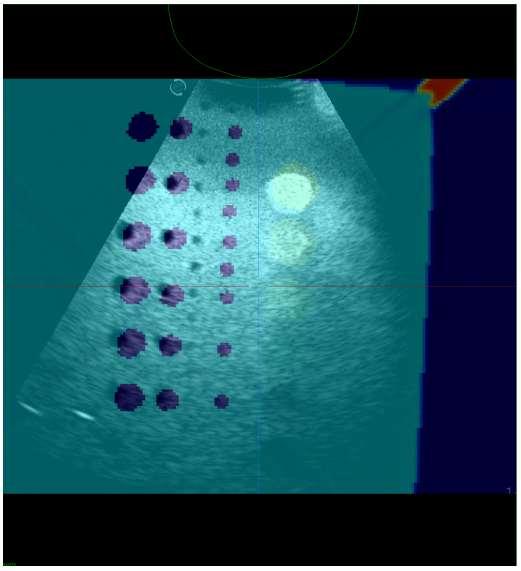
(a) CT-Bild mit nachträglich eingefügten Strukturen in der entsprechenden Schallebene



(b) Mit Tracking-Information verknüpftes Ultraschallbild




(c) Schematische Darstellung der inneren Strukturen des Phantoms



(d) Überlagerte Darstellung in der Anwendung

Abbildung 4.10: Abbildung (a) zeigt das farbcodierte CT-Bild entsprechend der Schallebene des Ultraschallbildes in (b). Abbildung (c) zeigt die Strukturen des Phantoms und (d) die überlagerte Darstellung.

sich zumindest explizit auf die Ultraschallbilder und die Repräsentation der Tracking-Werkzeuge. Für die Automation mit dem Roboter stehen vordefinierte Scanbewegungen zur Verfügung. Das 3D Ultraschallbild lässt sich aber auch komplett Freihand erstellen. Für diesen Fall lässt sich wie oben beschrieben die Anzahl der Bilder und die Aufnahmefrequenz einstellen. Nachteil des Freihand-3D-Ultraschallbildes ist, das ggf. Lücken entstehen.
Abbildung 4.11 zeigt wie ein erstelltes Ultraschallbild eines Probanden aussehen kann. Zu sehen ist ein Ausschnitt der Leber. Die Scanbreite ist 4 cm. Für diese Aufnahme wurden 360 Bilder verwendet. Das entsprach einer Aufnahmezeit von 12 s. Die Erstellung erfolgte automatisiert mit dem Roboter.Weitere 3D Aufnahmen befinden sich im Anhang B.
Durch eine parallelisierte und effizientere Berechnung des 3D Volumens ließ sich die Gesamtzeit des gezeigten Bildes auf dem verwendeten Gerät[1] auf insgesamt 16 s verkürzen (inkl. Aufnahmezeit). In einem klinischen Kontext bedeutet das, dass eine schnelle Akquisition und Darstellung des Untersuchungsbereichs möglich ist. Die erstellten Bilder sind nahezu ohne Lücken. Nach dem Erstellen kann das 3D Volumen in beliebigen Schichten betrachtet werden. Das 3D Ultraschallbild ist abgespeichert und kann für andere Zwecke verwendet werden. Es lassen sich so viele 3D Bilder erstellen wie es der Arbeitsspeicher des verwendeten Computers zulässt. Der Quellcode zum Berechnen des 3D Bildes kann in Anhang C.3 nachgeschaut werden.

## 4.5 Robotersteuerung

Die vorhandene Robotersteuerung wurde um zwei Funktionen erweitert. Es wurden Nachrichtenklassen implementiert und der bestehende Observer genutzt um ein Roboter-Tracking-System zu realisieren. Der Roboter kann nun seine aktuelle Position zur Verfügung stellen. Das kann sowohl der Flange des Roboters sein, als auch eine der Basen der montierten Werkzeuge. Die Steuerung schickt die aktuelle Position mit 30 Hz, was dem verwendeten optischen Tracking-System entspricht, an das MITK-Plugin. Es kann nun sowohl die Achsstellung als auch die Position des Roboters quasi parallel übertragen werden. Zweite Erweiterung war die Realisierung und Verbesserung von Scanverfahren zum Erstellen eines 3D Ultraschallbildes.

### 4.5.1 Scanverfahren

Hauptergebnis der Robotersteuerung sind die implementierten automatisierten Scanfunktionen. Diese erlauben es einen Bereich abzuscannen. Die Atemkompensation wird über den gesamten Scanvorgang und danach beibehalten. Es wurden drei Scanverfahren implementiert. Es wurden dabei nicht alle angegebenen Einstellmöglichkeiten in MITK später auch genutzt:

[1] Mac Book Pro 15 2017, virtuelle Maschine Windows 8, 8 GB Ram, 4 Prozessorkerne

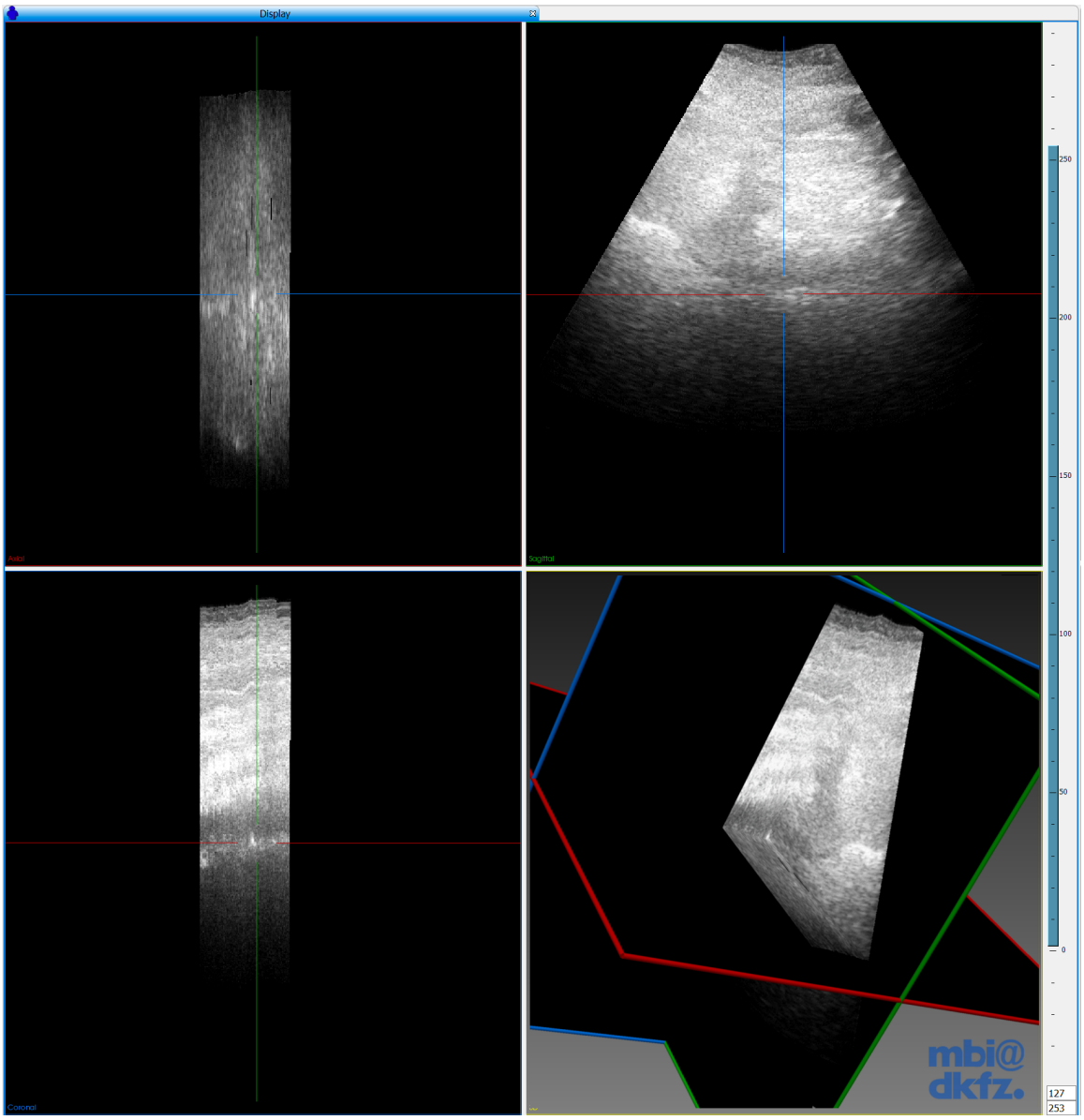



Abbildung 4.11: 3D Ultraschallbild Ansicht im MITK

- **Linear-Scan**: Bei diesem Scan verfährt der Roboter im Werkzeugkoordinatensystem linear entlang der x-Achse. Die zu scannende Strecke lässt sich definieren. Die Position vor dem Scannen definiert den Mittelpunkt der Strecke. Zum Scannen wird dann zuerst in die eine Richtung gescannt, dann zurück und über die Mitte und zuletzt erneut in die erste Richtung zurück zum Ausgangspunkt. Für ein lückenloses Bild kann die Geschwindigkeit der Bewegung angepasst werden. Die Kompensation erfolgt parallel und erlaubt es dem Roboter in Schallrichtung der Probandenoberfläche zu folgen. Abbildung 4.11 zeigt die Standard Scanfunktion. Eine Abbildung eines Linear-Scans von 13 cm Länge befindet sich in Anhang B.3.
- **Schwenk-Scan**: Beim Schwenk-Scan ist der Winkel das Scans einstellbar. Das erste Bild liegt auch hierbei in der Mitte. Eine Abbildung hiervon befindet sich in Anhang B.2.
- **Rotations-Scan**: Durch Drehen des Schallkopfes lässt sich ein kegelförmiges Bild erzeugen. Der Roboter ist hierbei so programmiert, dass er den Schallkopf einmal um 180° dreht. Die Startposition entspricht damit nicht der Endposition. Eine Abbildung hiervon befindet sich in Anhang B.1.

### 4.5.2 Verfeinerung und Tests

Die entwickelten Scanverfahren wurden mithilfe von Tests auf ihre reale Anwendbarkeit hin überprüft. Ergebnis dieser Tests ist, dass sich für eine robotergestützte 3D-Ultraschallbildgebung der Linear-Scan am besten eignet. Für diesen wurde eine ideale Scangeschwindigkeit von ca. 3 mm/s herausgefunden. Bei den anderen Verfahren zeigten die Tests, dass für ein lückenloses Bild deutlich mehr Zeit benötigt wird als beim linearen Scan. Weiter kann es bei Drehbewegungen eher zu problematischen Achsstellungen des Roboters kommen. Diese sind in sofern gefährlich, dass z.B. die implementierte Sicherheitsbewegung nicht möglich sein kann. Insbesondere beim Drehen des Schallkopfes um 180° ist die Ausgangsposition des Schallkopfes von zentraler Bedeutung. Je nach Position des Roboters ist diese Funktion nicht möglich.

## 4.6 Joystick-Module

Bereits in der Ausgangslage der Arbeit konnte der Joystick in einem Plugin verwendet werden. Neues Ergebnis dieses Moduls ist die allgemeinere Einbindung des Joysticks. Das bedeutet, er kann jetzt aus verschiedenen Plugins heraus verwendet werden. Durch die Vererbung der *mitk::BaseData*-Klasse können z.B. auch Filteroperationen zur Tastenzuweisung implementiert werden. Weiter ist ein Konzept erschaffen, das es erlaubt auch andere Gamecontroller auf die selbe Art und Weise einzubinden.

### 4.6.1 Aufbau

Das Modul setzt sich aus folgenden Klassen und Submodulen zusammen:

- **Modul DeviceRegisty**: Modul zur Aktivierung des *JoystickService* zu Beginn der Anwendung.
  - **JoystickActivator**: Registriert den Service bei Start der Anwendung.
- **JoystickData**: Repräsentiert die Daten eines Gamecontrollers. In diesem Fall die des Joysticks.
- **JoystickSource**: Quelle die als Ausgang *JoystickData* liefert und dazu die Daten des realen Gerätes lädt.
- **JoystickService**: Service der die *JoystickSource* verwaltet.

### 4.6.2 Joystick Data Source

Die *JoystickDataSource* erlaubt es den Joystick als normale Datenquelle zu verwenden. Die Quelle liefert dabei die Daten des Joysticks als *JoystickData*. Prinzipiell lässt sich die Quelle auch erweitern um mehrere Gamecontroller und auch andere Gamecontroller zu verwenden. Es lassen sich nun Filteroperationen verwenden um die Daten umzuwandeln, z.B. der in 4.7 erstellte Filter für die Benutzeroberfläche. Bei ersten Tests mit einem Arzt, war z.B. der Wunsch, dass je nach Position des Joysticks, die Achsen anders zugewiesen werden. Dies lässt sich nun realisieren.

## 4.7 Joystick-Plugin

An der Joystick Benutzeroberfläche hat sich nichts geändert. Eine Abbildung der Benutzeroberfläche befindet sich in Anhang (A.2). Es werden wie vorher die aktuell gedrückten Knöpfe angezeigt. Neues Ergebnis ist hierbei die veränderte Implementierung unter der Haube. Durch das neu erstellte Joystick-Modul musste ein neues Plugin erstellt werden. Da es nur eine kleine Teilaufgabe war wird nachfolgend nur der Aufbau kurz beschrieben. Das erstellte Plugin bietet nun aber eine saubere Schnittstelle um eine komplexere Nutzung von Gamecontrollern zu ermöglichen.

### 4.7.1 Aufbau

- **hhn_mi_robotproject_gui_qt_joystickview_Activator**: Standard Activatorklasse.
- **JoystickDataToGUIFilter**: Neu erstellter Filter zum Darstellen der Joystick-Werte.
- **JoystickView**: Verwaltungsklasse der Benutzerschnittstelle.

- **JoystickViewVontrols.ui**: Grafische Benutzerschnittstelle.

# 5 Diskussion

Problem bei der Bestrahlung von Tumoren im Abdomen ist, dass durch indizierte Bewegungen, wie z.B. Atmen, innere Strukturen ihre Position verändern und nicht mehr der Planungssituation entsprechen. Um sicherzustellen das genügend Dosis appliziert wird, werden größere Margins bestrahlt, mit höheren Nebenwirkungen für den Patienten. In verschiedenen Studien und Versuchen (Kapitel 2) konnte gezeigt werden, dass durch eine Positionskontrolle mittels Ultraschall eine präzisere Bestrahlung möglich wäre. Ultraschall erfordert einen kontinuierlichen Anpressdruck an die Patientenoberfläche, eine starre Halterung erfüllt diese Anforderung nicht und schränkt die möglichen Einstrahlrichtungen ein. In eigenen Vorarbeiten konnte bereits gezeigt werden wie sich ein Roboter für diese Aufgabe nutzen und in ein medizinisches Bildverarbeitungsprogramm integrieren lässt. Damit anhand der Ultraschallbilder zukünftig überhaupt eine Aussage über die Position von inneren Strukturen getroffen werden kann, müssen diese in einem Referenzsystem zugeordnet werden. Erst danach lässt sich, z.B. durch ein registriertes Planungs-CT, die reale Situation mit der Planungssituation vergleichen und anschließend eine Anpassung der Bestrahlung umsetzen.
Ziel dieser Arbeit war die Weiterführung der bisherigen Vorleistungen und die Erweiterung dieser um die Zuordnung der Ultraschallbilder in einem Referenez-Koordinatensystem. Es sollte dabei ein Gesamtkonzept entwickelt werden, dass es zukünftig ermöglicht eine ultraschallbasierte und robotergestützte Lokalisationskontrolle von Tumoren während der Strahlentherapie durchführen zu können. Dazu gehört die Integration der verschiedenen Systeme (Roboter, Ultraschall, usw.) in einer Applikation, um dem Nutzer eine kompakte Anwendung zu geben, die sich auch praktisch nutzen lässt. So gehört neben der Zuordnung der Bilder auch eine entsprechende Darstellung dazu.
Ergebnis dieser Arbeit bildet eine auf dem medizinischen Bildverarbeitungsprogramm MITK (2.5) basierte Applikation, die Funktionen für die oben genannten Ziele besitzt. Es konnten erfolgreich alle Hardewarekomponenten und bestehende Funktionen integriert werden. Die Applikation bietet so die Möglichkeit ein Tracking-System mit einem Ultraschallgerät zu verknüpfen. Die geforderte Zuordnung der Position des Bildes wurde damit erfolgreich realisiert und diese Bilder lassen sich in Echtzeit, in der Anwendung darstellen. Zusätzlich lassen sich Planungsdaten registrieren und überlagert mit dem Ultraschallbild darstellen. Weiter wurde eine Möglichkeit geschaffen um mit Hilfe des Roboters und einem 2D Ultraschallkopf ein automatisiertes 3D Ultraschallbild zu erzeugen. Die entwickelte Applikation ist klar strukturiert und lässt sich zukünftig für weiter benötigte Funktionen erweitern. Es wurde auf eine klare Benutzersteuerung geachtet und zeitkritische Methoden effizient implementiert, dazu gehört z.B. die Rekonstruktion eines 3D Ultraschallvolumens. Somit wurde neben dem umgesetzten Konzept eine praxisnahe und tatsächlich verwendbare Applikation geschaffen, die anhand von mehreren Tests überprüft wurde. Gerade im Umgang mit dem Roboter wurden neue Bewegungen getestet und für eine reale Verwendung optimiert. Beispiel hierfür ist die Möglichkeit nun automatisierte lückenfreie 3D Ultraschallbilder mit bis zu 1800 verwendeten Einzelbildern eines Probanden zu erstellen.
Zum Erreichen der Ziele wurden mehrere kleinere Komponenten entwickelt. In ei-

nem ersten Schritt wurde dabei eine Möglichkeit geschaffen den Roboter als Tracking-System nutzen zu können. Für die punktbasierte Registrierung der Planungsdaten ist die Positionierung des Roboters zu ungenau. Es wurde daher zusätzlich ein optisches Tracking-System verwendet. Um die Vorteile beider Systeme nutzen zu können wurde ein Filter entwickelt, der die Tracking-Systeme zu einem verknüpft. Zur Entwicklung der Positionszuweisung des Ultraschallbildes wurde ebenfalls ein passender Filter erstellt. Dieser wird auch zur Rekonstruktion eines 3D Ultraschallbildes verwendet.

## 5.1 Entwicklung eines Gesamtkonzept

Die Entwicklung eines Systems zur robotergestützten und ultraschallbasierten Lokalisationskontrolle ist keine Trivialität. Es gibt bereits mehrere Ansätze, die Teilbereiche betrachten und lösen. Es fehlt aber an einem gesamtheitlichen Konzept, dass es erlaubt zukünftig alle Bereiche miteinander zu verknüpfen. Das ist aber für eine reale Anwendung später unerlässlich. Damit möglichst viele existierende Komponenten wiederverwendet werden, wurde zu Beginn dieser Arbeit eine Evaluation der Ausgangslage (3.1) erstellt. Ergebnis dieser war, dass die bestehende Integration des Roboters erweitert werden soll. Für die Referenzierung der Position des Ultraschallbildes würde das sogenannte PLUS-Toolkit eine Realisierung bieten. Es ist eine Library, die Algorithmen hierfür besitzt und sich mit dem MITK verbinden lässt. Es würde nahe liegen dieses zu verwenden, aus den in Abschnitt 3.1.4 genanten Gründen wurde sich aber gegen eine Implementierung mit diesem Toolkit entschieden. Dazu zählt vor allem die steigende Komplexität des MITK-Projekts und der hohe Initialisierungsaufwand. Alternativ zu diesem Ansatz wurde eine eigene Implementierung realisiert, die bereits vorhandene Funktionen für 2D Ultraschall und Tracking-Systeme (2.5.3) des MITK nutzt. Hierzu wurden fehlende Komponenten, wie passende Filter und Darstellungsmöglichkeiten entwickelt. Zusammen mit der erweiterten Integration des Roboters als Tracking-System und zur Automation des 3D Ultraschalls bilden diese Punkte den Kern der neu entwickelten Applikation.

### 5.1.1 Erstellung der Applikation

Wie bereits erwähnt existierte bereits eine Einbindung des Roboters in das MITK. Diese bestand aus einem einzigen Plugin. Die darin verwendete Architektur erlaubte es nicht den Roboter auch aus anderen Plugins heraus zu verwenden und erschwerte es neue Funktionen zu erstellen. Damit bei der steigenden Komplexität der Gesamtanwendung die einzelnen Funktionen klarer gekapselt werden und neue Komponenten einfach dazukommen können wurde das bisherige Plugin in ein komplettes MITK-Projekt umgewandelt. Ein Projekt setzt sich aus Modulen und Plugins zusammen und erlaubt eine parallele Entwicklung der eigenen Anwendung zum eigentlichen MITK. Für den Roboter wurde so ein eigenes Modul geschaffen. Dieses beinhaltet eine Serviceklasse, die Funktionen des Roboters pluginübergreifend möglich macht. Es erlaubt

zukünftig auch anderen Nutzern einen Zugang zum Roboter. Die eigentliche Steuerung erfolgt weiterhin aus einem Plugin mit Benutzeroberfläche heraus. Dieses verwaltet den Roboter und gibt allgemeine Funktionen frei oder sperrt diese, z.B aus Sicherheitsaspekten. Im Rahmen dieser Umstellung bekam der für die Steuerung verwendete Joystick ebenfalls ein eigenes Modul, welches nun auch anderen Nutzern zur Verfügung steht. Wichtiger ist das neu erstellte Ultraschall-Tracking-Plugin es bietet die neuen Methoden für die Ultraschallbildgebung. Dazu gehört die Verknüpfung von Tracking-Systemen, die Verknüpfung von verschiedenen Tracking-System und Ultraschallgerät sowie die Darstellung des Ultraschallbildes in 3D, als auch Methoden zur 3D Ultraschallbildrekonstruktion. Diese werden in den weiteren Abschnitten ausführlich behandelt.

### 5.1.2 Erstellung und Kalibrierung von Werkzeugen

Neben der eigentlichen Applikation gehört auch die Kalibrierung der Tracking-Werkzeuge zu dem Gesamtkonzept. Ohne eine entsprechende Kalibrierung und deren Nachweis kann nicht sicher gesagt werden, dass die umgesetzte Methodik funktioniert. Für das optische Tracking-System wurden die Ergebnisse von Lissek [12] übernommen. Der FRE betrug dabei 1,27 mm. Für das Werkzeug wurde eine entsprechende .rom-Datei angelegt und die Kalibrierung dort hinterlegt. Mit Hilfe des optischen Tracking-Systems wurde das Roboter-Tracking-System, mit den robotereigenen Messmethoden, kalibriert. Beide Tracking-Systeme sollen später den gleichen Referenzpunkt liefern. Der Berechnungsfehler des Roboter-Tracking-Systems lag bei 0,99 mm. Zur Klärung ob sich die beiden Tracking-Systeme kombinieren lassen und ob man anhand des einen Tracking-Systems die Werte des anderen berechnen kann wurde ein Versuch durchgeführt. Es wurden hierzu 500 Messungen verglichen, jeweils die original Tracking-Daten des einen Systems mit den aus dem zweiten System berechneten. Im Median weicht die Position um 1,25 mm ab. Die Orientierung in Quaternionendarstellung ist im zweistelligen Nachkommastellenbereich genau (siehe auch Ergebnis 4.1.1). Die Kalibrierung ist daher für erste Tests ausreichend genau, jedoch sollte für eine spätere Verwendung eine präzisere Kalibrierung durchgeführt werden. Die Tracking-Systeme liefern beide die Position der Ultraschallkopfspitze, genau in der Mitte horizontal und vertikal am oberen Rand des eigentlichen Bildes. Dadurch können die verschiedenen Einstellmöglichkeiten des Ultraschallgeräts, wie z.B. Tiefe oder Winkel, berücksichtigt werden und benötigen nicht für jede Einstellung ein neues Werkzeug.

## 5.2 Nutzung des Roboters im Kontext der Aufgabe

Wie bereits mehrfach gezeigt ist der Roboter für die Gesamtaufgabe essentiell. Die Vorarbeiten beinhalten mit der Integration auch mehrere Bewegungsfunktionen. Neben Funktionen zum Positionieren des Roboters spielt die entwickelte Atemkompensationsbewegung und die Sicherheit eine zentrale Rolle. Für die neuen Anforderungen

wie automatische Scanfunktionen muss ebenfalls die Atemkompensation verwendet werden und die Sicherheit gewährleistet sein. Die auf Seiten der Robotersteuerung implementierten Funktionen erlaubten eine Erweiterung um die neuen Bewegungen und die Verwendung als Tracking-System, ohne eine Änderung der darin verwendeten Architektur.

### 5.2.1 Roboter-Tracking-System

In der Literatur wird für ein beliebiges Tracking-System gerne OpenIGTLink als Nachrichtenprotokoll benutzt [38]. Dieses bietet Nachrichtenklassen für medizinische Geräte und ist in vielen medizinischen Bildverarbeitungsprogrammen integriert, auch im MITK. Eine allgemeine Variante den Roboter als Tracking-System zu verwenden ist einen Server in der Robotersteuerung zu implementieren der Positionsnachrichten über das Nachrichtenprotokoll schickt. Dieser Ansatz funktionierte jedoch aufgrund der älteren Java-Version der Nachrichtenschnittstelle auf Seiten der Robotersteuerung nicht. Es wurde daher die bestehende Verbindung mit dem Roboter genutzt und ein eigenes Tracking-System entsprechend der vorhandenen MITK-Klassen erstellt. Der Nutzen und die Verwendung für den Anwender ist dabei der gleiche, jedoch würde die erste Variante einen wichtigen Vorteil bieten. Die Position des Roboters könnte von verschiedenen Anwendungen und unabhängig von der Roboterapplikation verwendet werden. So könnte z.B. der Linearbeschleuniger die Position des Roboters erfragen um Kollisionen mit diesem auszuschließen. Zukünftig muss daher geschaut werden ob eine Umstellung der Implementierung möglich wird.

### 5.2.2 Verknüpfung von Tracking-Systemen

Wie bereits erwähnt ist das Handführen des Roboters für eine punktbasierte Registrierung zu ungenau. Aus diesem Grund wurde ein optisches Tracking-System zur Registrierung von Planungsdaten verwendet. Dieses hat jedoch den Nachteil, dass eine Sichtverbindung notwendig ist. Durch die Bewegung der Gantry und des Roboters muss eine solche nicht gegeben sein. Die Möglichkeit beide Systeme kombiniert nutzen zu können ist somit wichtig. Im Rahmen der Arbeit wurde ein Filter erstellt, der anhand der Daten von beiden Tracking-Systemen eine Transformation zwischen diesen berechnen kann und die Werte des einen aus den Werten des anderen bestimmt (Ergebnis hierzu befindet sich in Abschnitt 4.2.3).

### 5.2.3 Neu implementierte Bewegungsabläufe

Die Robotersteuerung wurde um Bewegungsbefehle für die Erstellung eines 3D Ultraschallbildes erweitert. Für diese wurde die bestehende Atemkompensationsbewegung um weitere kontrollierte Bewegungsabläufe erweitert. Dazu gehört eine lineare Verfahrensart, das Schwenken des Schallkopfes und das Drehen um die Schallkopfachse. Für

diese Bewegungen wurden ideale Parameter für die Verfahrensgeschwindigkeit, Winkel oder Distanzen anhand von iterativen Tests entwickelt (3.9.2). Für den Anwender wurden drei vordefinierte Bewegungen hinterlegt, es lassen sich aber noch beliebig viele andere definieren und je nach Anwendungsfall verwenden. Bei der Drehung des Schallkopfes um dessen eigene Achse (Ergebnis ein 3D Kegel/Zylinder) besteht die Gefahr, dass der Roboter an die Randbereiche seiner Bewegungsmöglichkeit kommt. Gerade bei einer schlechten Ausgangslage besteht Gefahr das Singularitäten[1] auftreten. Es wird daher empfohlen diese Bewegung trotz der prinzipiellen Möglichkeit nicht zu verwenden. Ansonsten konnte in Tests die Verwendbarkeit der neuen Bewegungen auch am Probanden gezeigt werden. Die so aufgenommenen 3D Bilder enthalten kaum bis gar keine Lücken.

## 5.3 Positionszuweisung von Ultraschallbildern

Wie bereits mehrfach erwähnt ist die Positionszuweisung des Ultraschallkopfes essentiell. In dieser Arbeit konnte ein Filter erstellt werden, der diese Aufgaben übernimmt. Weiter wurde eine Möglichkeit geschaffen dem Anwender das Bild auch in 3D darzustellen. Abschließend konnte außerdem gezeigt werden, dass eine überlagerte Darstellung mit Planungsdaten möglich ist.

### 5.3.1 Entwickelte Filter

Der entwickelte Filter ist ein einfacher Tracking-Daten zu Tracking-Daten-Filter, da es keine Überklasse für einen kombinierter Filter aus Bild und Tracking-Daten gibt. Er reicht dabei die Positionsinformation einfach weiter. Zum Setzen der Positionsinformation des Bildes besitzt der Filter weiter einen Eingang für eine allgemeine Bildquelle. Jedes Mal wenn neue Tracking-Daten an den Eingängen verfügbar sind, werden die darin enthaltenen Positionen mit einer Kalibrierungsmatrix verrechnet und dem parallel von der Bildquelle geladenen Bild übergeben. Die Kalibrierungsmatrix ist hier notwendig um das Bild mit dessen Koordinatensystem in das Refernzkoordinatensystem der Tracking-Daten zu transformieren, da diese die Spitze des Ultraschallkopfes liefern und nicht den Eckpunkt des Bildes. An einem weiteren Ausgang kann das so verknüpfte Bild abgefragt werden (Das Ergebnis hierzu steht in Abschnitt 4.2.4).
Der Filter lässt sich direkt mit dem Filter zum Kombinieren von zwei Tracking-Systemen verknüpfen. Es kann dadurch eine kleine Pipeline aufgebaut werden. Das so verknüpfte Bild lässt sich anschließend allgemein verwenden und es besteht die Möglichkeit Bildverarbeitungen oder Segmentierungen darauf durchzuführen. Die Voxel des Bildes enthalten nun, wie gewünscht, die Zuordnung in das Referenzsystem des Tracking-Systems.

[1] Singularität bedeutet, zwei Achsen stehen kollinear zueinander. Dadurch benötigt eine Achse eine unendliche Geschwindigkeit um die Bewegung ausführen zu können

### 5.3.2 Darstellung des Bildes

Es ist möglich das Ultraschallbild in den Standardansichten des MITK darzustellen. Dazu gehört insbesondere die 3D Darstellung. In dieser können sowohl das Ultraschallbild, als auch die Tracking-Werkzeuge, als auch die überlagerten Planungsdaten gleichzeitig dargestellt werden. Dem Nutzer kann so der reale Aufbau in der Anzeige des MITK sichtbar gemacht werden und das Nutzerfeedback verbessern. Damit die Anzeige in den Ansichten möglich war musste ein Bug, der entsteht wenn ein Bild in zwei oder drei Schichten rechtwinklig zu den Standardansichten liegt, behoben werden. Hierzu wurde eine Änderung am Core-Module vorgenommen entsprechend eines vorhanden Fixes[2]. Da dieses noch nicht im aktuellen Master-Branch des MITK eingepflegt ist muss geschaut werden ob dieser Bug ggf. anders behoben wird oder andere Nachteile durch die verwendete Variante entstehen. Ergebnisse hierzu befinden sich in Abschnitt 4.4.3.

### 5.3.3 Überlagerungsmöglichkeit mit Planungs-Daten

Durch die bereits vorhandenen IGT-Plugins (2.5.3) kann eine punktbasierte Registrierung mit dem optischen Tracking-System erfolgen. Durch die Art der Darstellung des Ultraschallbildes wird automatisch die entsprechende Schnittebene durch das Planungs-Bild dargestellt. Es lassen sich beide Bilder durch Setzen von Transparenzwerten überlagert darstellen. Hier wäre es wünschenswert, wenn noch andere Darstellungsformen möglich wären. Die Überlagerung konnte in Tests verwendet werden und zeigte, dass die Überlagerung als solche funktioniert und dass das Tracking-Werkzeug eine präzisere Kalibrierung benötigt um eine vollständige Überlagerung herzustellen. Verwendet wurde hierzu ein Ultraschallphantom und dessen CT-Daten. Ergebnisse hierzu stehen in Abschnitt 4.4.4.

## 5.4 Rekonstruktion von 3D Ultraschallbildern

Wie bereits in einigen Abschnitten erwähnt wurde eine Möglichkeit geschaffen aus mehreren 2D Ultraschallbildern und deren Positionsinformation ein 3D Ultraschallbild zu berechnen. Dieses 3D Ultraschallbild kann später z.B. für einen schnellen Überblick genutzt werden. Ein weiterer Grund für die Verwendung in dieser Arbeit war es einen Nachweis für die richtige Zuordnung der Ultraschallbilder zu erhalten. Ein 3D Ultraschallbild bietet hierzu ein gutes visuelles Feedback. In späteren Anwendungen könnte auch ein 3D Ultraschallkopf verwendet werden. Ein solcher war nicht vorhanden und es besteht für einen solchen auch keine Integration in das MITK. Weiter ist auch nicht klar ob es überhaupt ein 3D Bild für die Bestimmung der inneren Bewegungen benötigt. Ansonsten erfolgt die Zuordnung identisch wie der entwickelte Ansatz. Um

[2] https://github.com/MITK/MITK/pull/203/

dennoch die Vorteile eines solchen 3D Ultraschallbildes zu nutzen wurden Funktionen für die Rekonstruktion implementiert. Diese wurden effizient umgesetzt und es kann so auf dem getesteten System[3] innerhalb von 16 s ein 4 cm großer Linear-Scan (roboterunterstützt) mit 360 verwendeten Bildern, dargestellt werden. Weiter können bis zu 1800 Bilder für ein einzelnes Volumen verwendet werden. Theoretisch lassen sich noch mehr verwenden, da aber bereits bei 1800 Bildern die Aufnahmezeit bei 1 min liegt, entsteht durch noch mehr Bilder kein praktischer Nutzen mehr. Ergebnisse hierzu stehen in Abschnitt 4.4.5.

### 5.4.1 Beschleunigung des Algorithmus

Damit ein 3D Bild später einen Gewinn bedeutet und einen praktischen Nutzen hat, muss sich dieses schnell berechnen lassen. Der Algorithmus zur Rekonstruktion wurde daher mit der Akquisition parallelisiert. Das bedeutet, dass während der Aufnahme der Bilder parallel die Eintragung der Bilder in das Volumen erfolgt. Weiter wurden die Ränder des Ultraschallbildes abgeschnitten um unnötige Pixel nicht zu berücksichtigen. Die Berechnungszeit dauert dadurch im Schnitt nur noch 1/3 länger als die Akquisitionszeit. In einem ersten naiven Ansatz erfolgte das Berechnen erst nach dem Akquirieren. Hier dauerte der gesamte Prozess 3 mal so lang wie die Aufnahmezeit. Sollten weitere Beschleunigungen notwendig sein, muss die Grafikkarte verwendet werden.

### 5.4.2 Automatisierbarkeit mit dem Roboter

Ein klassisches Freihand 3D Ultraschall hat den Nachteil das Lücken entstehen können. Wie bereits in Abschnitt 5.2.3 erwähnt wurde bietet der Roboter nun Scanfunktionen an. Diese nutzen definierte Bewegungen, die keine oder nur minimale Lücken erzeugen. Der Roboter wird hierzu über dessen Service angesprochen. Die Aufnahme im erstellten Ultraschall-Plugin und die Bewegung des Roboters laufen synchron ab. Die beiden Funktionen laufen an sich jedoch unabhängig voneinander ab, damit der Roboter im Fehlerfall die Bewegung sofort abbricht und nicht durch die Bildgebung beeinflusst wird.

## 5.5 Ausblick

Die in der Arbeit gestellten Ziele wurden erreicht. Es konnte ein Konzept zur robotergestützten und ultraschallbasierten Lagekontrolle erstellt werden. Wichtige Teilaspekte dieses Problems wurden implementiert und realisiert. Dazu gehört die Zuordnung des Ultraschallbildes in ein Referenzkoordinatensystem. Zukünftig lassen sich detektierte Strukturen in den Bildern eindeutig im Raum zuordnen. Es ist dabei möglich

[3]Mac Book Pro 2017 15 Windows 8 Virtuelle Maschine 8 GB Arbeitsspeicher und 4 Prozessorkerne

den Bezug zu Planungsdaten herzustellen und eine überlagerte Darstellung des Ultraschallbildes mit den registrierten Planungsdaten darzustellen. Die Basis für zukünftige Algorithmen, die Strukturen und deren Bewegung erkennen, ist gelegt. Die mit Positionsdaten versehenen Bilder lassen sich für diesen Zweck weiterverwenden. Die Applikation baut auf bestehenden Funktionen auf und erlaubt seinerseits die neuen Funktionen weiter zu nutzen. Zukünftig kann so Schritt für Schritt die Anwendung erweitert werden.
Neben der Bildzuordnung liefert die Arbeit eine Applikation die sowohl Funktionen für die Bildverarbeitung als auch die Integration des Roboters beinhaltet. Dieser lässt sich für die Bildakquisition, z.B. für Scanverfahren nutzen und erlaubt es einem Benutzer die Schallkopfposition auch nachträglich zu verändern. Weiter kann die Achsstellung verändert werden um z.B. der Gantry des Bestrahlungssystems auszuweichen. Zukünftig kann die Bildverarbeitung direkt mit der Robotersteuerung verknüpft werden um z.B. eine Zielstruktur automatisiert durch den Roboter verfolgen zu lassen. Dadurch würde neben der Atemkompensation ein weiterer Schritt zur vollständigen Automation eines solchen Gesamtsystems und die dadurch einfachere klinische Nutzung unternommen werden. In naher Zukunft müssen weiter ausführliche Tests zur Sicherheit und Stabilität der Applikation und insbesondere des Roboters gemacht werden. Zahlreiche Tests wurden zwar bereits durchgeführt, für klinische Versuche sind diese jedoch noch lange nicht ausreichend.
Am Ende des Weges lässt sich folgende Anwendung vorstellen: Der Patient kommt normal zur Bestrahlung. Der Medizinisch Technische Assistent (MTA) platziert den Roboter an einer definierten Stelle. Der Roboter schwenkt den Ultraschallkopf selbständig und registriert anhand des Ultraschallbildes die Position des Patienten. Automatisiert fährt er dann für die Bestrahlung eine günstige Ausgangslage, um innere Bewegungen zu detektieren, an. Die Bestrahlung als solche nutzt die Bildinformation und wandelt die Planungssituation entsprechend ab und steuert in Echtzeit den MLC. Nach der Behandlung ist eindeutig klar welche Struktur welche Dosis appliziert bekommen hat.
Bis hier hin ist es aber noch ein weiter Weg, für welchen die Arbeit einen kleinen aber entscheidenden Beitrag leisten konnte.

# 6 Literaturverzeichnis


[1] M. F. Fast, T. P. O'Shea, S. Nill, U. Oelfke, and E. J. Harris, "First evaluation of the feasibility of mlc tracking using ultrasound motion estimation," *Medical Physics*, vol. 43, no. 8Part1, pp. 4628–4633, 2016.

[2] P. K. Seitz, "Konzeption und entwicklung einer applikation zur robotergestützten ultraschallbildgebung." `urn:nbn:de:bsz:840-opus4-1018`, 2015.

[3] P. K. Seitz and R. Bendl, "Robotergestützte ultraschallbildgebung zur lagekontrolle von zielgewebe während der strahlentherapie," in *Bildverarbeitung für die Medizin 2017* (K. H. Maier-Hein, geb. Fritzsche, T. M. Deserno, geb. Lehmann, H. Handels, and T. Tolxdorff, eds.), (Berlin, Heidelberg), pp. 287–292, Springer Berlin Heidelberg, 2017.

[4] T. Falco, G. Shenouda, C. Kaufmann, I. Belanger, C. Procaccini, C. Charrois, and M. Evans, "Ultrasound imaging for external-beam prostate treatment setup and dosimetric verification," *Medical dosimetry : official journal of the American Association of Medical Dosimetrists*, vol. 27, no. 4, p. 271?273, 2002.

[5] A. M. Priester, S. Natarajan, and M. O. Culjat, "Robotic ultrasound systems in medicine," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 60, pp. 507–523, March 2013.

[6] S. E. Salcudean, G. Bell, S. Bachmann, W. H. Zhu, P. Abolmaesumi, and P. D. Lawrence, *Robot-Assisted Diagnostic Ultrasound – Design and Feasibility Experiments*, pp. 1062–1071. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999.

[7] C. Graumann, B. Fuerst, C. Hennersperger, F. Bork, and N. Navab, "Robotic ultrasound trajectory planning for volume of interest coverage," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 736–741, May 2016.

[8] T. K. Adebar, O. Mohareri, and S. E. Salcudean, "Instrument-based calibration and remote control of intraoperative ultrasound for robot-assisted surgery," 06 2012.

[9] I. Kuhlemann, R. Bruder, F. Ernst, and A. Schweikard, "We-g-brf-09: Force- and image-adaptive strategies for robotised placement of 4d ultrasound probes," *Medical Physics*, vol. 41, no. 6Part30, pp. 523–523, 2014.

[10] S. Tauscher, J. Tokuda, G. Schreiber, T. Neff, N. Hata, and T. Ortmaier, "Openigtlink interface for state control and visualisation of a robot for image-guided therapy systems," *International Journal of Computer Assisted Radiology and Surgery*, vol. 10, no. 3, pp. 285–292, 2015.

[11] K. März, A. M. Franz, B. Stieltjes, J. Iszatt, A. Seitel, B. Radeleff, H.-P. Meinzer, and L. Maier-Hein, "Mobile em field generator for ultrasound guided navigated needle insertions," in *Information Processing in Computer-Assisted Interventions* (D. Barratt, S. Cotin, G. Fichtinger, P. Jannin, and N. Navab, eds.), (Berlin, Heidelberg), pp. 71–80, Springer Berlin Heidelberg, 2013.

[12] C. Lissek, "3d-ultraschall beim mitk." `urn:nbn:de:bsz:840-opus4-1232`, 2017.

[13] D. M. Muratore and J. Galloway, Robert L., "Beam calibration without a phantom for creating a 3-d freehand ultrasound system," *Ultrasound in Medicine and Biology*, vol. 27, pp. 1557–1566, 2018/02/26 2001.

[14] J. Boda-Heggemann, P. Mennemeyer, H. Wertz, N. Riesenacker, B. Küpper, F. Lohr, and F. Wenz, "Accuracy of ultrasound-based image guidance for daily positioning of the upper abdomen: An online comparison with cone beam ct," *International Journal of Radiation Oncology Biology Physics*, vol. 74, no. 3, pp. 892–897, 2009.

[15] M. Fuss, B. J. Salter, S. X. Cavanaugh, C. Fuss, A. Sadeghi, C. D. Fuller, A. Ameduri, J. M. Hevezi, T. S. Herman, and J. Thomas, Charles R., "Daily ultrasound-based image-guided targeting for radiotherapy of upper abdominal malignancies," *International Journal of Radiation Oncology Biology Physics*, vol. 59, no. 4, pp. 1245–1256, 2003.

[16] A. Hsu, N. R. Miller, P. M. Evans, J. C. Bamber, and S. Webb, "Feasibility of using ultrasound for real-time tracking during radiotherapy," *Medical Physics*, vol. 32, no. 6Part1, pp. 1500–1512, 2005.

[17] D. A. Kuban, L. Dong, R. Cheung, E. Strom, and R. De Crevoisier *Seminars in Radiation Oncology*, 2005.

[18] K. M. Langen, J. Pouliot, C. Anezinos, M. Aubin, A. R. Gottschalk, I.-C. Hsu, D. Lowther, Y.-M. Liu, K. Shinohara, L. J. Verhey, V. Weinberg, and I. Roach, M, "Evaluation of ultrasound-based prostate localization for image-guided radiotherapy," *International Journal of Radiation Oncology Biology Physics*, vol. 57, pp. 635–644, 2018/02/17.

[19] J. Wu, O. Dandekar, D. Nazareth, P. Lei, W. D'Souza, and R. Shekhar, "Effect of ultrasound probe on dose delivery during real-time ultrasound-guided tumor tracking," in *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 3799–3802, 2006.

[20] S. Ipsen, R. Bruder, R. O?Brien, P. J. Keall, A. Schweikard, and P. R. Poulsen, "Online 4d ultrasound guidance for real-time motion compensation by mlc tracking," *Medical Physics*, vol. 43, no. 10, 2016.

[21] J. Schlosser, K. Salisbury, and D. Hristov, "Telerobotic system concept for real-time soft-tissue imaging during radiotherapy beam delivery," *Medical Physics*, vol. 37, no. 12, pp. 6357–6367, 2010.

[22] C. Dekomien, "Registrierung von 3d-ultraschall mit mrt- und ct-daten für die navigierte chirurgie." `http://www-brs.ub.ruhr-uni-bochum.de/netahtml/HSS/Diss/DekomienClaudia/diss.pdf`, 2013.

[23] KUKA, *KUKA Sunrise.OS 1.3 KUKA Sunrise.Workebench 1.3*, 31.7.2014.

[24] KUKA, *KUKA Sunrise.Connectivity Servoing 1.7*, 23.4.2015.

[25] KUKA, *KUKA Sunrise Cabinet*, 29.7.2014.

[26] KUKA, *LBR iiwa*, 31.7.2014.

[27] KUKA, *Sicherheit LBR iwa*, 29.7.2014.

[28] "Mitk homepage." `http://mitk.org/wiki/MITK`.

[29] "Itk homepage." `https://itk.org`, 1 2017.

[30] "Vtk homepage." `https://www.vtk.org`.

[31] "Mitk tutorial." `http://docs.mitk.org/2016.11/TutorialPage.html`.

[32] "Mitk developer manual." `http://docs.mitk.org/nightly/DeveloperManualPortal.html`.

[33] "Ultrasound plugin." `http://docs.mitk.org/nightly/org_mitk_gui_qt_ultrasound.html`, 2.2.2018.

[34] "Igt tracking toolbox plugin." `http://docs.mitk.org/nightly/org_mitk_views_igttrackingtoolbox.html`, 2.2.2018.

[35] "Igt navigation tool manager plugin, year = 2.2.2018, howpublished = `http://docs.mitk.org/nightlyorg_mitk_views_igtnavigationtoolmanager.html`,."

[36] "Igt registration view plugin." `http://docs.mitk.org/nightly/org_mitk_views_igtregistrationview.html`, 2.2.2018.

[37] "Measurement plugin." `http://docs.mitk.org/nightly/org_mitk_views_measurement.html`, 2.2.2018.

[38] M. Klemm, T. Kirchner, J. Gröhl, D. Cheray, M. Nolden, A. Seitel, H. Hoppe, L. Maier-Hein, and A. M. Franz, "Mitk-openigtlink for combining open-source toolkits in real-time computer-assisted interventions," *International Journal of Computer Assisted Radiology and Surgery*, vol. 12, pp. 351–361, Mar 2017.

[39] A. Lasso, T. Heffter, A. Rankin, C. Pinter, T. Ungi, and G. Fichtinger, "Plus: open-source toolkit for ultrasound-guided intervention systems," *IEEE transactions on bio-medical engineering*, vol. 61, pp. 2527–2537, 10 2014.

[40] GLFW Homepage, "GLFW Library." `http://www.glfw.org`, 6.2.2018.

# Anhang

# A Screenshots

## A.1 NDI 6D Architekt

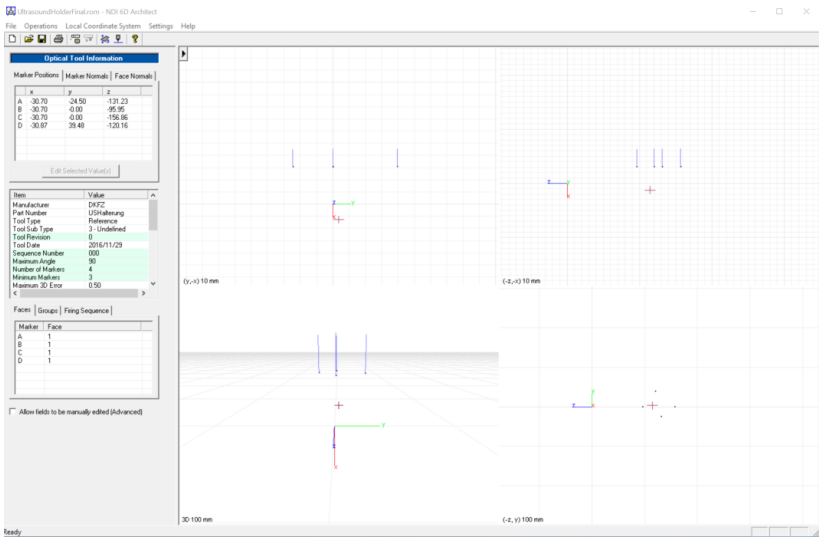



Abbildung A.1: NDI 6D Architekt: Der Screenshot zeigt eine Software zur Vermessung von optischen Tracking-Werkzeugen. Die Ansicht zeigt dabei die Marker und die zugehörige Basis.

## A.2 Joystick-Plugin

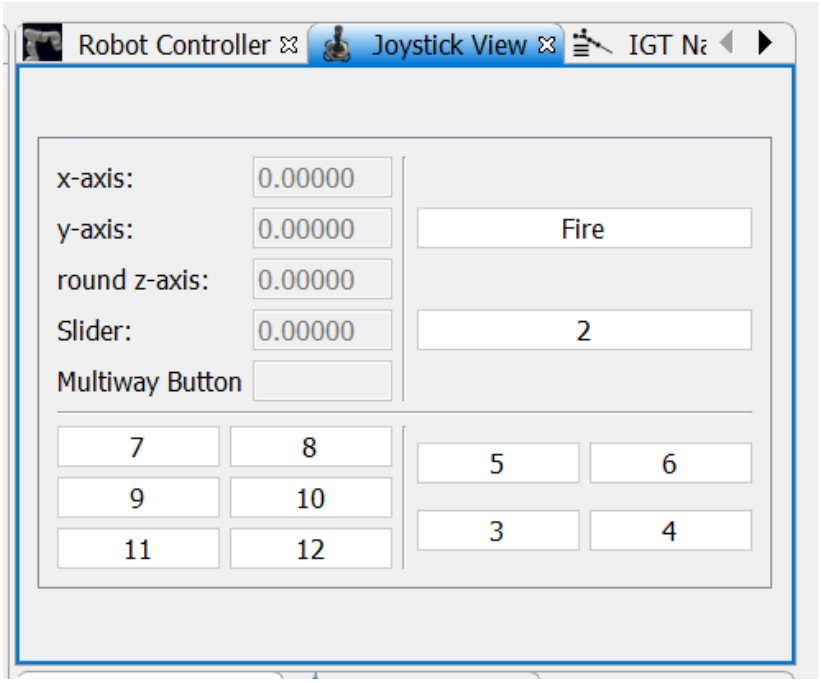



Abbildung A.2: Joystick- Benutzerschnitstelle. Es werden die aktuell gedrückten Knöpfe angezeigt und die entsprechenden Achswerte.

# B Weitere erstellte Ultraschallbilder

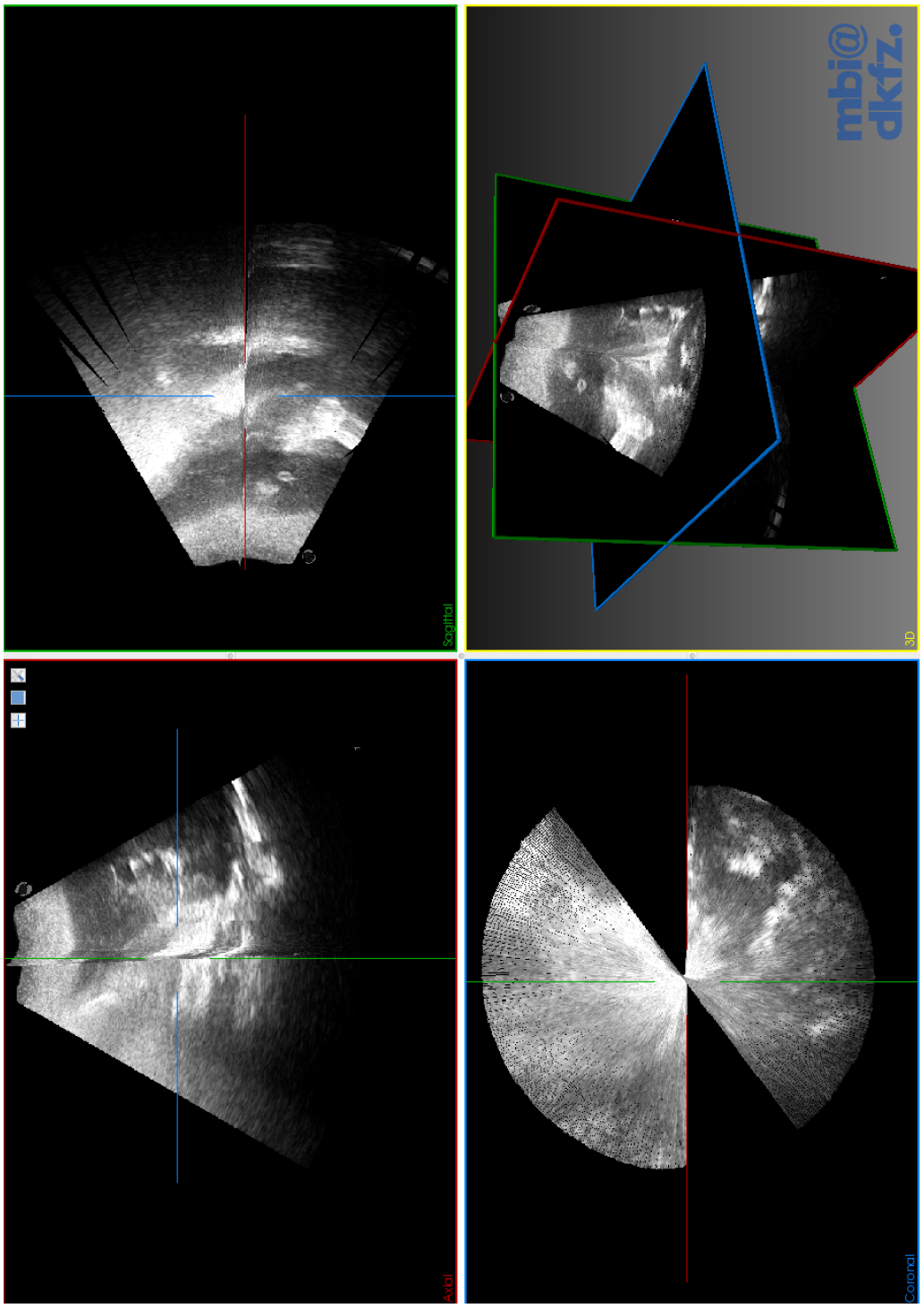



Abbildung B.1: 3D-Rotations-Ultraschallscan der Leber: Zum Erstellen eines lückenlosen Kegels wird zuviel Zeit benötigt. Es ist zu erkennen, dass an den Rändern bereits Lücken vorhanden sind. Aufnahmezeit 1 min 1800 Bilder. Das Bild wurde robotergestützt erstellt.

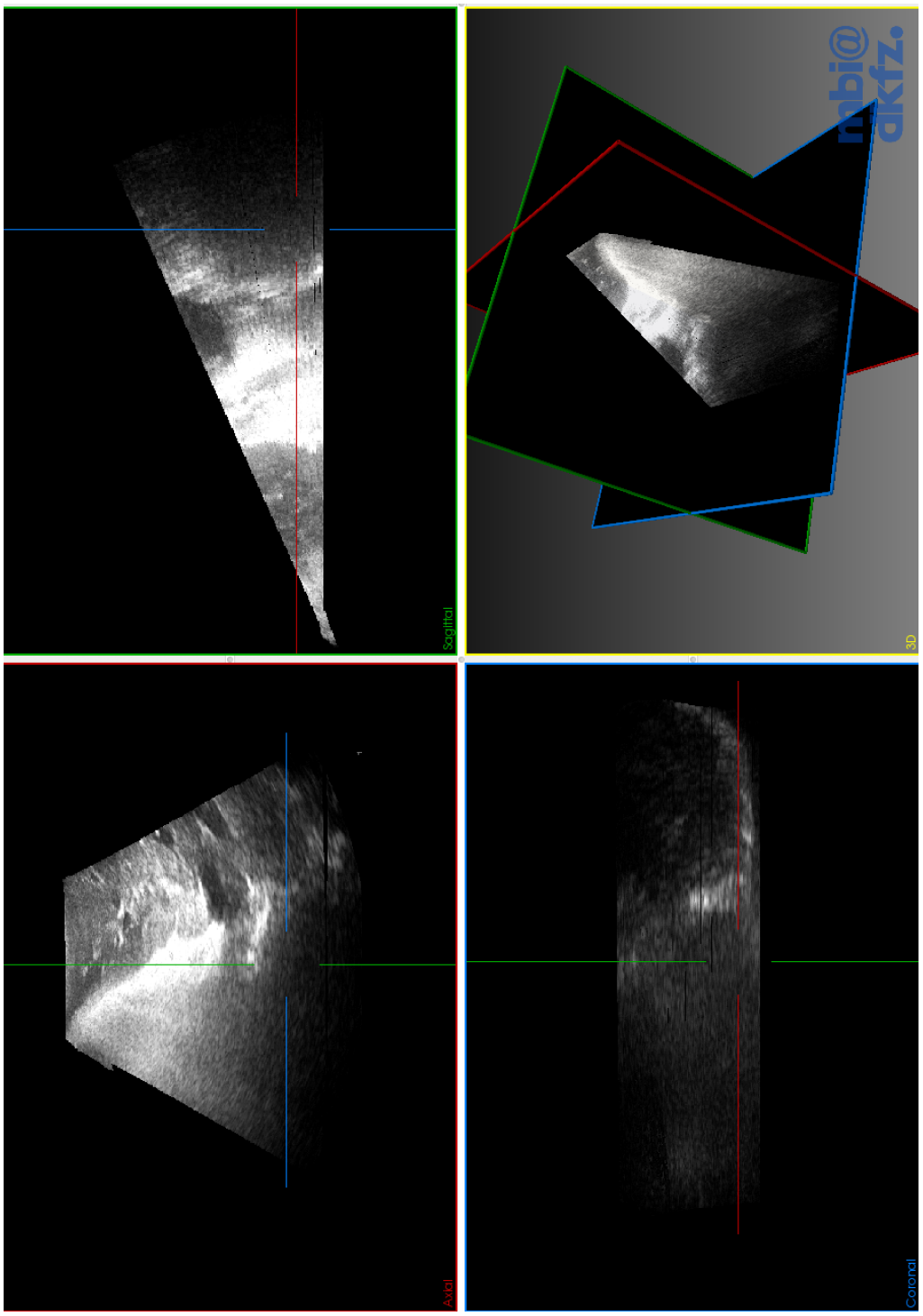



Abbildung B.2: 3D-Schwenk-Ultraschallscan der Leber: Es wurden 360 Bilder verwendet bei 12 s Aufnahmezeit. Das Bild wurde robotergestützt erstellt.

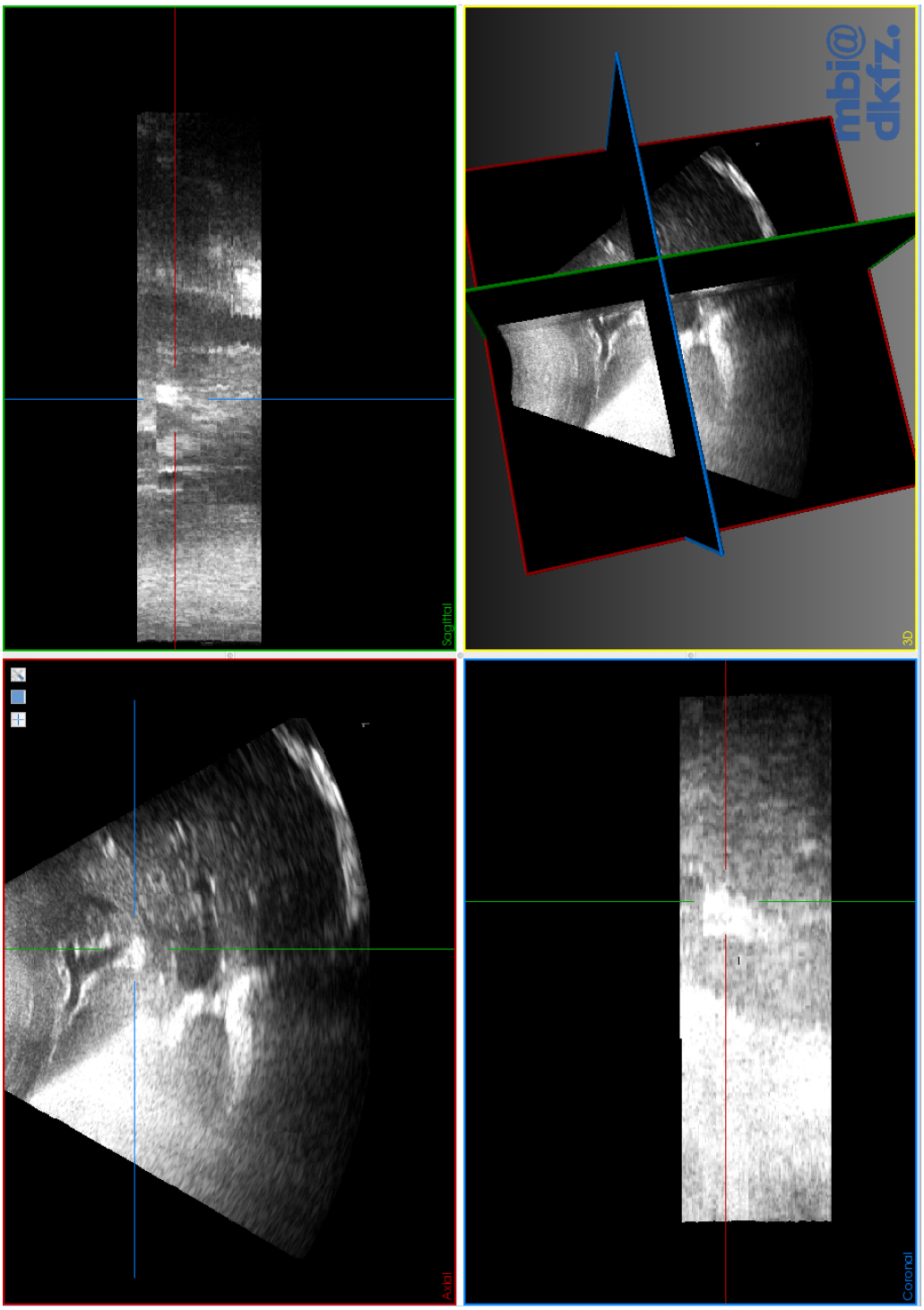



Abbildung B.3: 3D-Linear-Ultraschallscan der Leber: Das 3D Volumen wurde robotergestützt erstellt. Es wurden 360 Bilder bei 12 s Aufnahmezeit verwendet.

Abbildung B.4: 3D-Linear-Ultraschallscan der Leber: Das 3D Volumen entstand aus einem 13 cm Linear Scan. Verwendet wurden 1800 Bilder bei 1 min Aufnahmezeit. Der Roboter wurde verwendet. Die Aufnahme erfolgte jedoch nicht automatisch, sondern manuell mit dem Joystick, dadurch sind kleinere Lücken entstanden.

Abbildung B.5: 3D-Linear-Ultraschallscan mit starker Bewegung. Verwendet wurden 1800 Bilder bei 1 min Aufnahmezeit. Der Roboter wurde verwendet. Die Aufnahme erfolgte jedoch nicht automatisch, sondern manuell mit dem Joystick. Durch einen Lachanfall des Probanden sind die Schichten stark verrutscht. Das Bild spiegelt keine Anatomie mehr wieder, zeigt aber, dass der Roboter die Atem- und Bewegungskompensation erfolgreich ausführt und jederzeit eine Bildinformation verfügbar war.

# C Quellcode

## C.1 Tracking-Combine-Filter

```
1  void mitk::UltrasoundTrackingVisualizationFilter::GenerateData() {
2      mitk::Image::Pointer image = m_UsDevice->GetOutput()->Clone();
3      int i = 0;
4      mitk::NavigationData* output = this->GetOutput(i);
5      assert(output);
6      const mitk::NavigationData* input = this->GetInput(i);
7      assert(input);
8       output->Graft(input); // First, copy all information from input
            to output
9
10     mitk::AffineTransform3D::Pointer tDeviceAffinMatrix = input->
           GetAffineTransform3D();
11
12     vtkMatrix4x4 *a = UltrasoundTrackingVisualizationFilter::
           getVTKMatrix(tDeviceAffinMatrix);
13     vtkMatrix4x4 *result = vtkMatrix4x4::New();
14     vtkMatrix4x4::Multiply4x4(a, m_CalibrationMatrix, result);
15
16     image->GetGeometry()->
           SetIndexToWorldTransformByVtkMatrixWithoutChangingSpacing(
           result);
17     image->SetSpacing(m_Spacing);
18     m_OutputImage = image;
19 }
```

Listing C.1: Generate Data Methode des Tracking-Combine-Filter. Die Daten des Tracking-Gerät werden unverändert übertragen. Die Position des Bildes wird anhand der Kalibrierungsmatrix und der Trackin-Daten gesetzt.

## C.2 Ultraschall-Tracking-Filter

```
1  void mitk::UltrasoundTrackingCombineFilter::GenerateData() {
2      int i = 0;
3      mitk::NavigationData* output0 = this->GetOutput(0);
4      assert(output0);
5      const mitk::NavigationData* input0 = this->GetInput(0);
6      assert(input0);
7
8      mitk::NavigationData* output1 = this->GetOutput(1);
9      assert(output1);
10     const mitk::NavigationData* input1 = this->GetInput(1);
11     assert(input1);
12     output0->Graft(input0); // First, copy all information from input
           to output
13     output1->Graft(input1);
14
15     if (!output0->IsDataValid()) {
16         mitk::AffineTransform3D::Pointer tDeviceAffinMatrix = input1->
               GetAffineTransform3D();
17
18         vtkMatrix4x4 *a = UltrasoundTrackingCombineFilter::getVTKMatrix
               (tDeviceAffinMatrix);
19
20         vtkMatrix4x4 *result = vtkMatrix4x4::New();
21         vtkMatrix4x4::Multiply4x4(m_CalibrationMatrix,a, result);
22
23         mitk::Point3D p;
24
25         p.SetElement(0,result->GetElement(0, 3));
26         p.SetElement(1,result->GetElement(1,3));
27         p.SetElement(2, result->GetElement(2, 3));
28         mitk::Quaternion q;
29
30           getQuaternion(q,result);
31
32         output0->SetPosition(p);
33         output0->SetOrientation(q);
34         output0->SetDataValid(true);
35     }
36 }
```

Listing C.2: Generate Data Methode des Ultraschall-Tracking-Filter. Sollten die Daten des Eingang eins nicht valide sein, wird der Eingang zwei für die Positionsbestimmung genutzt

## C.3 3D-Ultraschall

```
1  void ThreeDImageCalculaterThread::run() {
2      mitk::ProgressBar *p = mitk::ProgressBar::GetInstance();
3      while (m_Ultrasound3DImages.size() < 1) msleep(10);
4      mitk::Image::Pointer middleImage = m_Ultrasound3DImages[0];
5      vtkMatrix4x4 *middleImageTransformation = middleImage->GetGeometry
           ()->GetVtkMatrix();
6      int max = (middleImage->GetDimension(0) > middleImage->
           GetDimension(1)) ? middleImage->GetDimension(0) : middleImage->
           GetDimension(1);
7
8      unsigned int dimensions[] = { max + 40,max + 40,max + 40 };
9      mitk::Image::Pointer image3D = mitk::Image::New();
10     image3D->Initialize(middleImage->GetPixelType(), 3, dimensions);
11
12     const float originOld[] = { -20, -20,-max / 2 ,1 };
13     p->Progress(5);
14     float *originNew = middleImageTransformation->MultiplyFloatPoint(
           originOld);
15     vtkMatrix4x4 *origin3DImage = vtkMatrix4x4::New();
16     for (int i = 0; i < 4; ++i) {
17        for (int j = 0; j < 4; ++j) {
18           origin3DImage->SetElement(i, j, middleImageTransformation->
                 GetElement(i, j));
19        }
20     }
21     origin3DImage->SetElement(0, 3, originNew[0]);
22     origin3DImage->SetElement(1, 3, originNew[1]);
23     origin3DImage->SetElement(2, 3, originNew[2]);
24     image3D->GetGeometry()->SetIndexToWorldTransformByVtkMatrix(
           origin3DImage);
25
26      unsigned int dimensions[] = {  40, 40, 40 };
27      mitk::Image::Pointer image3D = mitk::Image::New();
28      image3D->Initialize(middleImage->GetPixelType(), 3, dimensions);
29      mitk::ImagePixelWriteAccessor<uchar, 3> writeAccess(image3D);
30
31     const int dimensionx = middleImage->GetDimension(0);
32     const int dimensiony = middleImage->GetDimension(1);
33      mitk::Point3D  worldcoords;
34      itk::Index<3> pixelindex = { { 0, 0, 0 } };
35      itk::Index<3> idx3 = { { 0, 0, 0 } };
36      myimage->GetGeometry()->IndexToWorld(idx3, worldcoords);
37      image3D->GetGeometry()->WorldToIndex(worldcoords, pixelindex);
38      writeAccess.SetPixelByIndexSafe(pixelindex, myimage->
            GetPixelValueByIndex(idx3));
39     emit(imageFinishSignal());
40 }
```

Listing C.3: Code zum Erstellen eines 3D Ultraschallbildes. Zuerst wird der 3D Würfel mit dessen Position und Orientierung bestimmt. Anschließend werden alle Bilder in diesesn Würfel eingetragen

# Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

(Ort, Datum) (Unterschrift)